7 Modelo de Análisis

Cuando ya se ha desarrollado y aceptado el modelo de requisitos se comienza el desarrollo del modelo de análisis. El objetivo del modelo de análisis es comprender y generar una arquitectura de objetos para el sistema en base a lo especificado en el modelo de requisitos. Durante esta etapa no se considera el ambiente de implementación, lo cual incluye al lenguaje de programación, manejador de base de datos, distribución o configuración de hardware, etc. En otras palabras el análisis pretende modelar el sistema bajo condiciones ideales, garantizando que la arquitectura de software resultante se suficientemente robusta y extensible para servir de base a la estructura lógica de la aplicación pero sin consideraciones relativas al entorno de implementación que es posible que cambien incluso radicalmente. Tarde o temprano el sistema tendrá que ser adaptado a las condiciones de implementación deseadas, algo que se hará durante el diseño, cuando todas las consideraciones que han sido descartadas durante el análisis sean consideradas. Es importante enfatizar que el modelo de análisis no es una reflexión del dominio del problema sino una representación de ésta adaptada a la aplicación particular. El modelo de análisis genera una representación conceptual del sistema, consistiendo de clases de objetos. Cada una de las clases de objetos contribuye de manera especial para lograr la robustez de la arquitectura, como se muestra conceptualmente en la Figura 7.1.

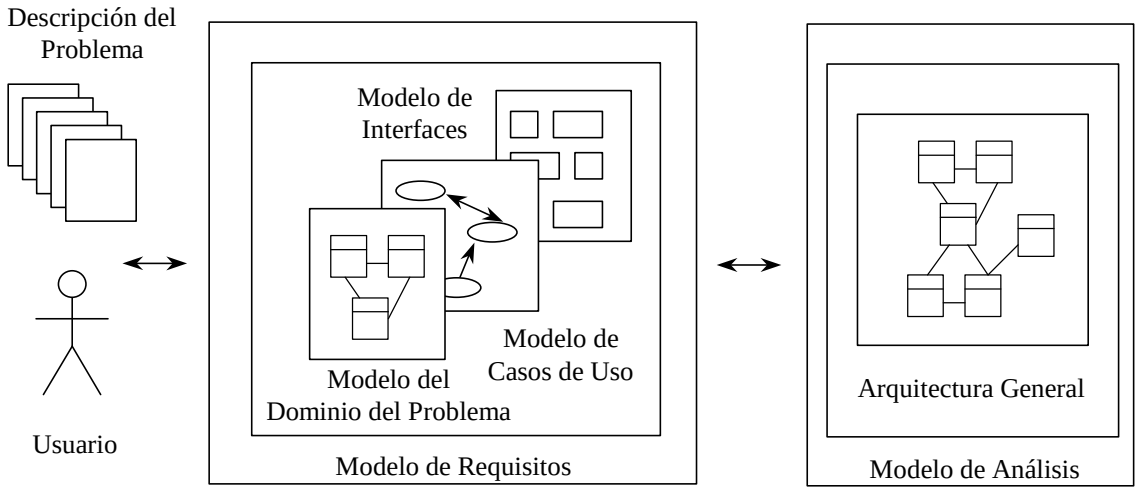

Figura 7.1 El diagrama muestra conceptualmente el modelo de análisis junto con la arquitectura general de objetos en relación al modelo de requisitos anteriormente desarrollado.

Una cualidad importante entre los diferentes modelos es el de la *rastreabilidad* (“traceability”) entre un modelo y otro, el cual relación sus distintos aspectos, tales como los objetos, sin importar el orden en que fueron desarrollados. Dado que la mayoría de los sistemas serán modificados a lo largo de su vida, si estos cambios emanan de cambios en los requisitos o respuestas a problemas, es necesario saber qué efectos tendrán estos sobre etapas posteriores incluyendo el código final.

7.1 Arquitectura de Clases

El modelo de análisis tiene como objetivo generar una arquitectura de objetos que sirva como base para el diseño posterior del sistema. Como se discutió en la introducción del libro, dependiendo del tipo de aplicación existen diversas arquitecturas que se pueden utilizar, siendo de nuestro interés aquellas arquitecturas especialmente diseñadas para el manejo de los sistemas de información, las cuales involucran ricas bordes de usuario y accesos a base de datos como aspectos fundamentales de la arquitectura.
En término de las propias arquitecturas, éstas se distinguen según la organización de la funcionalidad que ofrecen los objetos dentro de ellas o la *dimensión* de los objetos. Esta dimensión corresponde a los diferentes tipos de funcionalidad que manejan los objetos dentro la arquitectura. Por ejemplo, en el caso de funcionalidad para el manejo de bordes y base de datos, si existen tipos distintos de objetos para el manejo de cada una de estas por separado, entonces se considera que la arquitectura es de *dos* dimensiones. Por el contrario, si todos los objetos manejan de manera indistinta los dos tipos de funcionalidades, entonces se considera que la arquitectura es de *una* sóla dimensión.

Si aplicamos el concepto de dimensión a los métodos estructurados, podemos ver que estos consisten de dos dimensiones, correspondientes a funciones y datos. Las funciones representan un eje de comportamiento que no guarda información, mientras que los datos se ubican en un eje de información que no contiene comportamiento. En general, ejes de funcionalidad pueden corresponder a distintos tipos de funcionalidades, como se ve al contrastar funciones y datos versus manejo de bordes y bases de datos. Sin embargo, la pregunta más importante que uno se hace respecto al número y tipo de dimensiones es: ¿Si se diseña un sistema con múltiples dimensiones, se obtendría un sistema más robusto y sensible a modificaciones? Ante todo esta pregunta se relaciona con el concepto de modularidad, siendo muy aceptado que cuanto mayor sea la modularidad de un sistema mayor es su robustez y extensibilidad. La respuesta particular a la pregunta tiene que ver con qué tan independiente sea un eje de funcionalidad del otro, ya que en el caso de los métodos estructurados, usualmente se debe modificar las funciones cada vez que se modifica la estructura de información, lo cual no es algo deseable. Si logramos ejes de funcionalidad ortogonales, el efecto de cambios en una dimensión no debe afectar a las otras dimensiones. Y aunque estas dimensiones no son del todo ortogonales, si son lo suficientemente independientes se puede limitar el efecto de posibles cambios. En relación al número de dimensiones, esto depende de la funcionalidad que la arquitectura debe manejar, algo que a su vez depende del tipo de aplicación que se está desarrollando.
En le caso de los sistemas de información, uno de las tipos de arquitecturas mas importantes es la arquitectua *MVC – Modelo*, *Vista*, *Control* (*Model*, *View*, *Control*) popularizada por los ambientes de desarrollo de de Smalltalk. Esta arquitectura se basa en tres dimensiones principales: *Modelo* (*información*), *Vista* (*presentación*) y *Control* (*comportamiento*) como se muestra en la Figura 7.2.

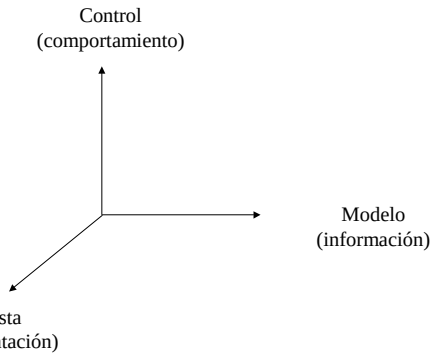

Figura 7.2 Diagrama de tres dimensiones correspondiente a la arquitectura *MVC – Modelo*, *Vista*, *Control*.
La vista o presentación de la información corresponde a las bordes que se le presentan al usuario para el manejo de la información, donde por lo general pueden existir múltiples vistas sobre un mismo modelo. Típicamente la información representa el dominio del problema y es almacenada en una base de datos. Por otro lado el control corresponde a la manipulación de la información a través de sus diversas presentaciones. Y aunque existe cierta dependencia entre estas tres dimensiones se considera que la manera de presentar la información es independiente de la propia información y de cómo esta se controla. Sin embargo, cada una de ellas probablemente experimente cambios a lo largo de la vida del sistema, donde el control es el más propenso a ser modificado, seguido de la vista y finalmente el modelo. En el modelo de análisis descrito aquí utilizaremos como base la arquitectura *MVC* para capturar estos tres aspectos de la funcionalidad, como se muestra en la Figura 7.3. Es importante notar la correspondencia con las tres dimensiones utilizadas durante el modelo de requisitos. La razón de ser de las tres dimensiones del modelo de requisitos realmente se deriva para lograr una rastreabilidad con la arquitectura que se desarrollará en el modelo de análisis.

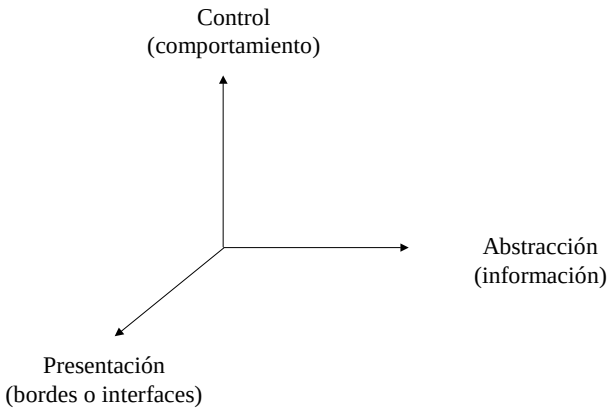

Figura 7.3 Diagrama de tres dimensiones correspondiente a la arquitectura del modelo de análisis basado en el modelo de casos de uso.

La arquitectura para el modelo de análisis será implementada mediante tres tipos o *estereotipos* de objetos como elementos básicos de desarrollo como veremos a continuación.

Clases con Estereotipos

El tipo de funcionalidad o "la razón de ser" de un objeto dentro de una arquitectura se le conoce como su *estereotipo*. Para los sistemas de información la arquitectura del sistema según nuestro modelo de análisis se basa en tres estereotipos básicos de objetos:

?? El estereotipo *entidad* ("entity" en inglés)para objetos que guarden información sobre el estado interno del sistema, a corto y largo plazo, correspondiente al dominio del problema. Todo comportamiento naturalmente acoplado con esta información también se incluye en los objeto entidad. Un ejemplo de un objeto entidad es un registro de usuario con sus datos y comportamiento asociados.

?? El estereotipo *interface* o *borde* ("boundary" en inglés) para objetos que implementen la presentación o vista correspondiente a las bordes del sistema hacia el mundo externo, para todo tipo de actores, no sólo usuarios humanos. Un ejemplo de un objeto borde es la funcionalidad de interface de usuario para insertar o modificar información sobre el registro de usuario.

?? El estereotipo *control* ("control" en inglés) para objetos que implementen el comportamiento o control especificando cuando y como el sistema cambia de estado, correspondiente a los casos de uso. Los objetos control modelan funcionalidad que no se liga naturalmente con ningún otro tipo de objeto, como el comportamiento que opera en varios objetos entidad a la vez, por ejemplo, hacer alguna computación y luego devolver el resultado a un objeto borde. Un ejemplo típico de objeto control es analizar el uso del sistema por parte de algún usuario registrado y presentar tal información posteriormente. Este comportamiento no le pertenece a ningún objeto entidad u objeto borde específico.

Nótese que no hay ninguna restricción a los diferentes estereotipos que puedan utilizarse, no solamente las tres anteriores. La notación de UML para un estereotipo se muestra en la Figura 7.4.

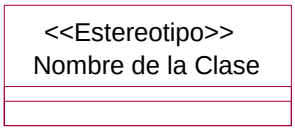

Figura 7.4 Diagrama de clase con estereotipo.

Los tres estereotipos correspondientes a las tres dimensiones para la arquitectura del modelo de análisis se muestra en la Figura 7.5.

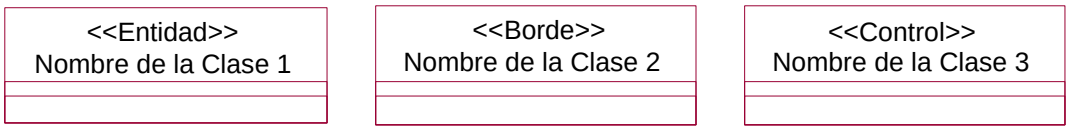

Figura 7.5 Diagrama de clase para los tres estereotipo.

Los tres estereotipos se muestran como íconos en la Figura 7.6. (Nótese que los estereotipos deben insertarse en inglés para que las herramientas CASE los reconozcan de tal manera.)

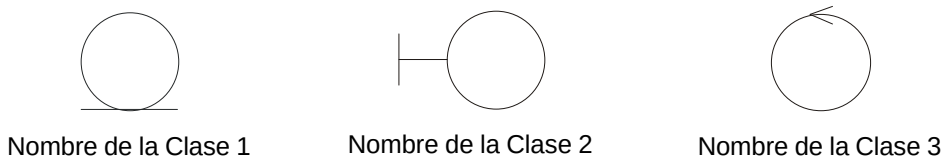

Figura 7.6 Diagrama de clase para los tres estereotipo.

Considerando que habrá interacción entre los diferentes tipos de objetos, existirá cierto traslape en la funcionalidad que los objetos ofrecen. Como se mencionó anteriormente, este traslape deberá minimizarse para asegurar una buena extensibilidad, donde típicamente, cada tipo de objeto captura por lo menos dos de las tres dimensiones. Sin embargo, cada uno de ellos tiene cierta inclinación hacia una de estas dos dimensiones, como se muestra en la Figura 7.7.

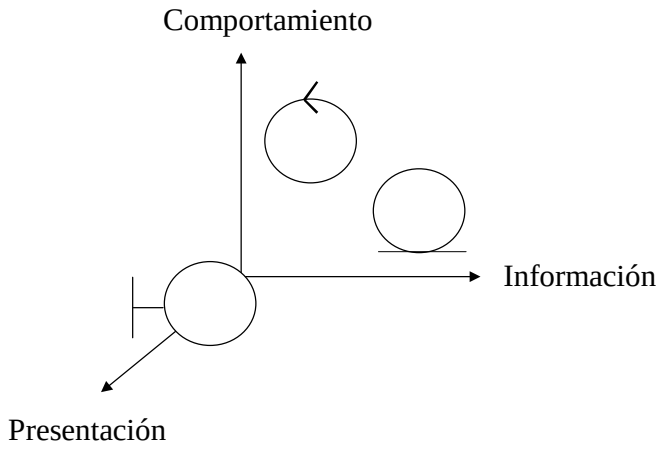

Figura 7.7 Diagrama mostrando traslape en los estereotipos de los objetos.

Clases para Casos de Uso

Cuando se trabaja en el desarrollo del modelo de análisis, normalmente se trabaja con un caso de uso a la vez. Para cada caso de uso se identifican los objetos necesarios para su implementación. Se identifican estos objetos según sus estereotipos para corresponder a la funcionalidad ofrecida en cada caso de uso. Se define explícitamente qué objeto es responsable de cual comportamiento dentro del caso de uso. Típicamente se toma un caso de uso y se comienza identificando los objetos borde necesarios, continuando con los objetos entidad y finalmente los objetos control. Este proceso se continúa a los demás casos de uso. Dado que los objetos son “ortogonales” a los casos de uso, en el sentido de que un objeto puede participar en varios casos de uso, este proceso es iterativo. Esto significa que cuando un conjunto de objetos ya existe, estos pueden modificarse para ajustarse al nuevo caso de uso. La meta es formar una arquitectura lo más estable posible, reutilizando el mayor número de objetos posible. De tal manera, la descripción original de los casos de uso se transforma a una descripción en base a los tres tipos de objetos, como se muestra en la Figura 7.8.

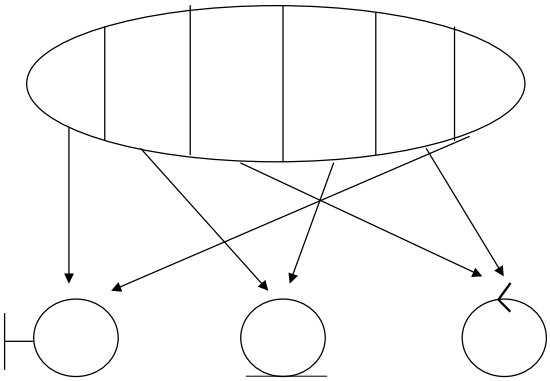

Figura 7.8 La funcionalidad de cada caso de uso es asignada a objetos distintos y de acuerdo a los estereotipos de dichos objetos.

Se parte el caso de uso de acuerdo a los siguientes principios:

?? La funcionalidad de los casos de uso que depende directamente de la interacción del sistema con el mundo externo se asigna a los objetos borde.

?? La funcionalidad relacionada con el almacenamiento y manejo de información del dominio del problema se asigna a los objetos entidad.

?? La funcionalidad específica a uno o varios casos de uso y que no se ponen naturalmente en ningún objeto borde o entidad se asigna a los objetos control. Típicamente se asigna a un sólo objeto control y si éste se vuelve muy complejo se asignan objetos control adicionales.

En las siguientes secciones identificamos las clases según sus estereotipos. El desafío para principal en dicho proceso es decidir cuantas y cuales clases deben deben asignarse por caso de uso.

7.2 Identificación de Clases según Estereotipos

Para llevar a cabo la transición del modelo de requisitos al modelo de análisis se deben identificar los objetos necesarios para implementar todos los casos de uso. La arquitectura de objetos debe considerar los tres tipos de estereotipos de objetos como se discutió anteriormente. Para lograr esto se debe identificar primero las clases borde, luego las entidad y finalmente las de control. En general, se desea asignar la funcionalidad más especializada correspondiente a la "política" de la aplicación a los objetos control, la cual depende y afecta al resto de los objetos. Por otro lado, los objetos entidad e borde deben contener funcionalidad más bien local limitando su efecto en los demás objetos. El trabajo del analista consiste en distribuir lo mejor posible el comportamiento especificado en el modelo de requisitos en los diferentes tipos de objetos de la arquitectura de análisis.

La asignación de funcionalidad es bastante difícil en la práctica afectando de gran manera la robustez y mantenimiento del sistema. Los buenos analistas consideran cambios potenciales al sistema a la hora de llevar a cabo este proceso.

En general, los cambios mas comunes a un sistema son los cambios en su funcionalidad e bordes. Cambios a las bordes deben afectar típicamente solo los objetos borde. Cambios a la funcionalidad son mas difíciles, ya que la funcionalidad puede abarcar todos los tipos de objetos. Si la funcionalidad esta ligada a la información existente, tales cambios afectan al objeto entidad representado esa información, o puede involucrar múltiples objetos incluyendo objetos control. Típicamente, esta funcionalidad se define en uno o varios casos de uso y se asigna a uno o varios objetos control.

A continuación describimos en más detalles el proceso de identificación de los tres tipos de objetos.

Borde

Toda la funcionalidad especificada en las descripciones de los casos de uso que depende directamente de los aspectos externos del sistema se ubica en los objetos de borde. Es a través de estos objetos que se comunican los actores con el sistema. La tarea de un clase borde es traducir los eventos generados por un actor en eventos comprendidos por el sistema, y traducir los eventos del sistema a una presentación comprensible por el actor. Las clases borde, en otras palabras, describen comunicación bidireccional entre el sistema y los actores.

Las clases borde son bastante fáciles de identificar, donde se cuenta con al menos tres estrategias:

1. Se pueden identificar en base a los actores.

2. Se pueden identificar en base a las descripciones de las borde del sistema que acompañan al modelo de requisitos.
3. Se pueden identificar en base a las descripciones de los casos de uso y extraer la funcionalidad que es específica a los bordes.

Comenzaremos utilizando la primera estrategia correspondiente a de actores. Cada actor concreto necesita su propia clase borde para su comunicación con el sistema. En muchos casos un actor puede necesitar de varios objetos borde. Es evidente que los objetos borde no son totalmente independientes de cada uno ya que deben saber de la existencia de los demás para poder resolver ciertas tareas. Por ejemplo para *Reservar* un *Asiento* en un *Vuelo* el usuario debe interactuar con las clases borde que a su vez se comunican con las clases borde que se comunican con la base de datos del sistema de reservaciones.

Una vez identificado los objetos borde es más fácil modificar posteriormente las clases borde de un sistema. Al tener todo lo relacionado a una clase borde en un objeto, cada cambio a la borde será local a ese objeto. Como los cambios a las bordes son muy comunes, es vital que estos sean extensibles.

Existen dos tipos diferentes de bordes a modelar, bordes a otros sistemas e bordes a los usuarios humanos.

?? En el caso de objetos borde que se comunican con otros sistemas, es muy común que la comunicación se describa mediante protocolos de comunicación. Los objetos borde pueden traducir las salidas del sistema a un protocolo de comunicación estandarizado, o simplemente enviar eventos producidos internamente sin conversiones complejas. Una ventaja de esto, es que si se cambia el protocolo, estos cambios serán locales al objeto borde. Un mayor problema ocurre cuando existen señales continuas del mundo externo, como en los sistemas de medición o control. Entonces los objetos borde deben muestrear la señal de entrada, o interrumpir cuando ciertos valores exceden un valor umbral, ya que internamente en el sistema sólo existe comunicación discreta mediante eventos. Los objetos borde deben entonces traducir la información continua a información discreta. Problemas de cuantificación pueden aparecer y deben ser resueltos.

?? En el caso de los objetos borde que se comunican con usuarios humanos, los objetos borde pueden ser complejos para modelar. Existen muchas técnicas diferentes para un buen diseño de bordes, como el diseño de Interfaces Gráficas de Usuario (GUI - Graphical User Interface), Sistemas de Manejo de Ventanas de Usuario (UIMS - User Interface Management Systems) y sistemas de Interface de Programación de Aplicación (API). Es fundamental que el usuario tenga una imagen lógica y coherente del sistema. En las aplicaciones interactivas es común que la borde de usuario sea una parte mayor (hasta 80%) de la aplicación completa.

Aunque cada tipo de objeto tiene un propósito distinto, es evidente que los objetos borde tienen como propósito principal las presentaciones. Sin embargo, también pueden manejar información y tener comportamiento. Cuánta información y comportamiento debe ligarse a un objeto borde debe decidirse de manera individual. En un extremo, el objeto borde solo envía el evento que recibe del actor a otros objetos en el sistema, sin participar activamente en el curso de eventos. En el otro extremo, el comportamiento del objeto borde es muy complejo donde la información se integra en el objeto borde y puede funcionar casi independiente de otros objetos.

Generalmente, el potencial para cambios debe afectar la decisión de qué comportamiento en el caso de uso debe ligarse a un objeto borde particular. Cualquier cambio en la funcionalidad directamente ligada a la borde debe ser local al objeto borde, mientras que otros cambios no deben afectarlo. Esto es algo que debe aprenderse y aplicarse en todas las actividades del modelado.

Para identificar qué parte del flujo de un caso de uso debe asignarse a los objetos borde, se debe analizar las interacciones entre los actores y los casos de uso. Esto significa buscar aspectos con una o más de las siguientes características:

?? Presentación de información al actor que requiera información de regreso.
?? Funcionalidad que cambie si cambia el comportamiento del actor.
?? Flujo de acción que dependa de un tipo de borde particular.

En el ejemplo del sistema de reservaciones de vuelo, cada uno de los actores concretos, *Cliente*, *Base de Datos de Registro* y *Base de Datos de Reserva*, necesita su propio objeto borde al sistema, como se muestra en la Figura 7.9. El *Usuario* necesita de las pantallas de presentación, mientras que la *Base de Datos de Registro* y *Base de Datos de Reservas* necesitan sus propias bordes para poder intercambiar información con el sistema.

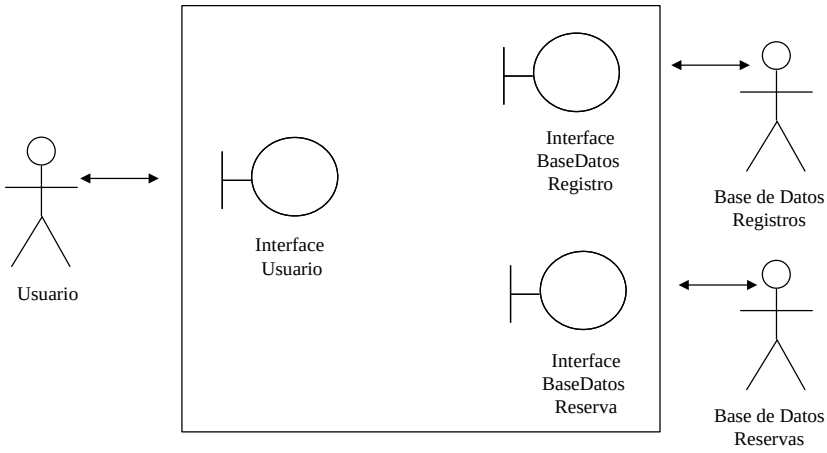

Figura 7.9 Clases borde para el sistema de reservaciones de vuelo identificados directamente de los actores.

Aunque estas tres clases borde son suficiente para interactuar con los actores, necesitamos incluir un número de clases borde adicionales correspondientes a cada pantalla que se le presenta al usuario, como se especificó inicialmente durante el modelo de requisitos. Por otro lado pueden haber clases bordes adicionales necesarias para manejos más especializados de las bases de datos, algo que postergaremos hasta el diseño. A continuación describimos las clases borde necesarias para cada caso de uso de acuerdo a la documentación generada durante el modelo de requisitos del capítulo anterior. Se requieren todas las pantallas con las cuales los casos de uso se relacionan. Nótese que a pesar de que existen múltiples ligas entres pantallas para simplificar la navegación, sólo se identifican como parte del caso de uso aquellas que se consideran "esenciales" para la ejecución del caso de uso.

?? *Validar Usuario*: Se interactúa con los actores Usuario y Base de Datos Registros a través de las clases borde *InterfaceUsuario* e *InterfaceBaseDatosRegistro*, respectivamente. Se utiliza únicamente la pantalla principal del sistema (*P-1*) para la validación de usuario. Por lo tanto se incluye únicamente la clase borde *PantallaPrincipal* además de las dos anteriores. Recuérdese que pantallas adicionales como las de mensajes o error no las estamos considerando aún para nuestro prototipo. En la Figura 7.12 se muestran las clases borde identificadas en este caso de uso.

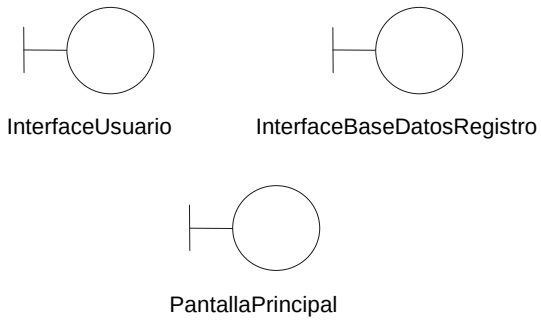

Figura 7.12 Clases borde identificadas del caso uso *Validar Usuario*.

?? *Ofrecer Servicios*: Este caso de uso utiliza únicamente la pantalla de servicios del sistema (*P-2*). Por lo tanto se incluye únicamente la clase borde *PantallaServicio*. Dado que se interactúa con el actor Usuario se incluye también la clase borde *InterfaceUsuario*. En la Figura 7.13 se muestran las clases borde identificadas en este caso de uso.

Figura 7.13 Clases borde identificadas del caso uso *Ofrecer Servicios*.

?? *Registrar Usuario*: Se interactúa con los actores Usuario y Base de Datos Registros a través de las clases borde *InterfaceUsuario* e *InterfaceBaseDatosRegistro*, respectivamente. Adicionalmente se deben incluir clases borde correspondientes a las pantallas propias de este caso de uso, que son las pantalla de registro de usuario por primera Vez (*P-3*) y de obtener registro (*P-4*). A las dos clases borde correspondientes las llamaremos

PantallaCrearRegUsuario y *PantallaObtenerRegUsuario*, respectivamente. Aunque la funcionalidad comienza en la pantalla principal del sistema (*P-1*) durante la validación de un usuario, esta validación se hace a través del caso de uso *Validar Usuario*, por lo cual esta funcionalidad no es incluida como parte de este caso de uso. En la Figura 7.14 se muestran las clases borde identificadas en este caso de uso.

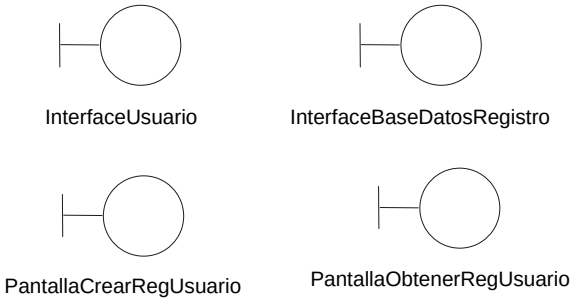

Figura 7.14 Clases borde identificadas del caso uso *Registrar Usuario*.

?? *Registrar Tarjeta*: Se interactúa con los actores Usuario y Base de Datos Registros a través de las clases borde *InterfaceUsuario* e *InterfaceBaseDatosRegistro*, respectivamente. Se utilizan las pantallas de registro de tarjeta por primera vez (*P-5*) y registro de tarjeta (*P-6*). A las dos clases borde correspondientes las llamaremos *PantallaCrearRegTarjeta* y *PantallaObtenerRegTarjeta*, respectivamente. En la Figura 7.15 se muestran las clases borde identificadas en este caso de uso.

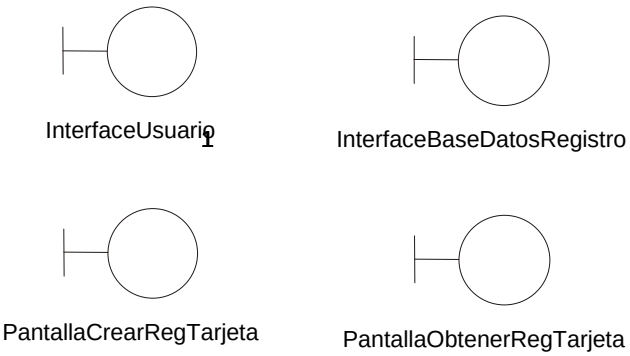

Figura 7.15 Clases borde identificadas del caso uso *Registrar Tarjeta*.

?? *Consultar Información*: Se interactúa con los actores Usuario y Base de Datos Reservas a través de las clases borde *InterfaceUsuario* e *InterfaceBaseDatosReserva*, respectivamente. Adicionalmente se deben incluir clases borde correspondientes a las pantallas propias de este caso de uso, que son las pantalla de selección de tipo de consulta (*P-7*), consulta de horarios de vuelos (*P-8*), resultado de consulta de horarios de vuelos (*P-9*), consulta de tarifas de vuelos (*P-10*), resultado de consulta de tarifas de vuelos (*P-11*), consulta de estado de vuelo (*P-12*) y resultado de consulta de estado de vuelo (*P-13*). A las clases borde correspondientes las llamaremos *PantallaConsultas, PantallaConsultaHorarios, PantallaResultadoHorarios, PantallaConsultaTarifas, PantallaResultadoTarifas, PantallaConsultaEstado* y *PantallaResultadoEstado*, respectivamente. En la Figura 7.16 se muestran las clases borde identificadas en este caso de uso.

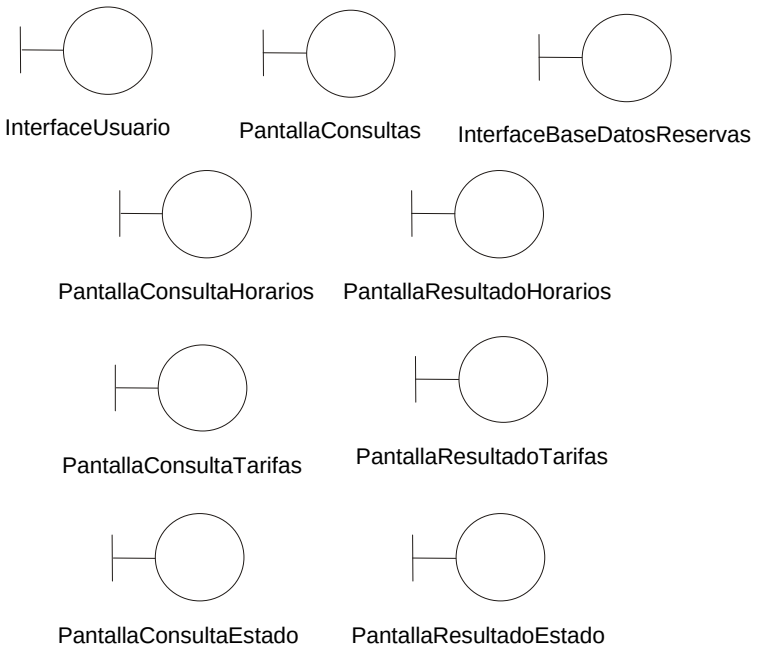

Figura 7.16 Clases borde identificadas del caso uso *Consultar Información*.

?? *Hacer Reservación*: Se interactúa con los actores Usuario y Base de Datos Reservas a través de las clases borde *InterfaceUsuario* e *InterfaceBaseDatosReserva*, respectivamente. Adicionalmente se deben incluir clases borde correspondientes a las pantallas propias de este caso de uso, que son la pantalla de inserción de clave de reserva (*P-14*), solicitud de reserva de vuelos (*P-15*) y récord de reserva de vuelos (*P-16*). A las clases borde correspondientes las llamaremos *PantallaClaveReservas*, *PantallaCrearReservaVuelos* y *PantallaRecordReservaVuelos*, respectivamente. En la Figura 7.17 se muestran las clases borde identificadas en este caso de uso.

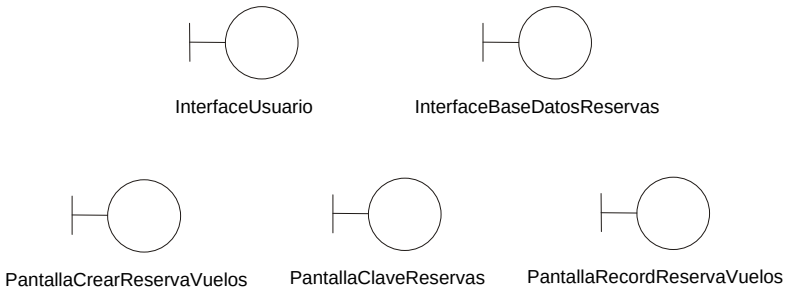

Figura 7.16 Clases borde identificadas del caso uso *Hacer Reservación*.

?? *Pagar Reservación*: Se interactúa con los actores Usuario y Base de Datos Reservas a través de las clases borde *InterfaceUsuario* e *InterfaceBaseDatosReservas*, respectivamente. Se utilizan las pantallas de pago de reserva de vuelos (*P-17*) y reembolso de reserva de vuelos (*P-18*). A las dos clases borde correspondientes las llamaremos *PantallaPagarRegTarjeta* y *PantallaReembolsarRegTarjeta*, respectivamente. En la Figura 7.18 se muestran las clases borde identificadas en este caso de uso.

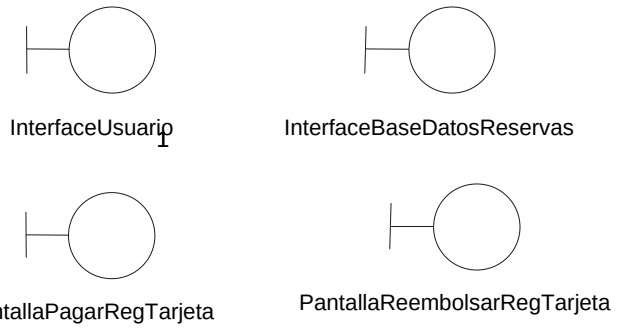

Figura 7.18 Clases borde identificadas del caso uso *Pagar Reservación*.

En la Tabla 7.1 se muestran el resumen los casos de uso identificados durante el modelo de requisitos junto con los actores y clases borde correspondientes.

Caso de Uso	Actores	Clases Borde
Validar Usuario	*Usuario, Base de Datos Registros*	*InterfaceUsuario, PantallaPrincipal, InterfaceBaseDatosRegistro*
Ofrecer Servicios	*Usuario*	*InterfaceUsuario, PantallaServicio*
Registrar Usuario	*Usuario, Base de Datos Registros*	*InterfaceUsuario, PantallaCrearRegUsuario, PantallaObtenerRegUsuario, InterfaceBaseDatosRegistro*
Registrar Tarjeta	*Usuario, Base de Datos Registros*	*InterfaceUsuario, PantallaCrearRegTarjeta, PantallaObtenerRegTarjeta, InterfaceBaseDatosRegistro*
Consultar Información	*Usuario, Base de Datos Reservas*	*InterfaceUsuario, PantallaConsultas, PantallaConsultaHorarios, PantallaResultadoHorarios, PantallaConsultaTarifas, PantallaResultadoTarifas, PantallaConsultaEstado, PantallaResultadoEstado, InterfaceBaseDatosReserva*
Hacer Reservación	*Usuario, Base de Datos Reservas*	*InterfaceUsuario, PantallaClaveReservas, PantallaCrearReservaVuelos, PantallaRecordReservaVuelos, InterfaceBaseDatosReserva*
Pagar Reservación	*Usuario, Base de Datos Reservas*	*InterfaceUsuario, PantallaPagarRegTarjeta, PantallaReembolsarRegTarjeta, InterfaceBaseDatosReserva*

Tabla 7.1 Relación entre casos de uso, actores y clases borde para el sistema de reservaciones de vuelo.

Entidad

Se utilizan objetos entidad para modelar la información que el sistema debe manejar a corto y largo plazo. La información a corto plazo existe por lo general durante la ejecución del caso de uso, mientras que la información a largo plazo sobrevive a los casos de uso, por lo cual es necesario guardar esta información en alguna base de datos. Adicionalmente, se debe incluir comportamiento para manejar la propia información local al objeto entidad.

Los objetos entidad se identifican en los casos de uso, donde la mayoría se identifican del modelo del dominio del problema en el modelo de requisitos. Objetos entidad adicionales pueden ser mas difíciles de encontrar. Es muy común que se identifiquen muchos objetos entidad, aunque se debe limitar estos objetos a los realmente necesarios para la aplicación, siendo esto lo más difícil del proceso de identificación. Es por lo tanto esencial trabajar de forma organizada cuando se modelan los objetos entidad. Las necesidades de los casos de uso deben ser las guías y solamente aquellos objetos entidad que puedan justificarse de la descripción del caso de uso deben ser incluidos.

No es siempre fácil decidir cuando cierta información debe ser modelada como un objeto entidad o como un atributo. Esto depende de cómo se usará la información, si ésta se maneja de forma separada, debe modelarse como un objeto entidad, mientras que la información que esta acoplada fuertemente a alguna otra información y nunca se usa por si misma debe modelarse como un atributo de un objeto entidad. Todo depende de cómo los casos de uso manejen la información. Cierta información puede modelarse como objeto entidad en un sistema, mientras que en otro sistema puede ser un atributo.

Es también difícil identificar qué operaciones y cuales atributos serán incluidos dentro de los objetos entidad. Dado que la única forma para manipular un objeto entidad es por medio de sus operaciones, las operaciones identificadas deben ser suficientes para manipular completamente al objeto entidad. La descripción detallada de los casos de uso es de nuevo un medio extremadamente valioso para encontrar las operaciones deseadas. El flujo

completo de eventos que se describe en los casos de uso, permite extraer las operaciones que conciernen a los objetos entidad.

Las operaciones asignadas a los objetos entidad pueden ser más o menos complejas. En el caso menos complejo un objeto entidad consta sólo de operaciones de acceso a los valores de los atributos. En el caso más complejo un objeto entidad puede tener flujos de eventos más allá de simples accesos a los valores de los atributos. Sea cual sea la complejidad de estas operaciones, el objetivo es que éstas sólo dependan y afecten información local. La siguiente es una lista de las operaciones típicas que deben ser ofrecidas por un objeto entidad:

?? Guardar y traer información
?? Comportamiento que debe modificarse si el objeto entidad cambia
?? Crear y remover el objeto entidad

Dada la complejidad de obtener operaciones, esto es un aspecto que se deja para la etapa de diseño, como se mencionó anteriormente.

Durante la identificación de objetos entidad, se encontrará que objetos similares aparecen en varios casos de uso. En tales circunstancias se debe verificar si deben ser los mismos objetos entidad o si deben haber objetos entidad separados. Incluso si los casos de uso no interactuan de la misma manera sobre los objetos, el objeto entidad puede ofrecer operaciones que satisfagan las necesidades de diversos casos de uso. Si se considera que dos objetos entidad representan un mismo objeto, las operaciones, atributos y asociaciones también tienen que integrarse.

De manera similar, se puede hacer una identificación preliminar de los atributos, sin embargo estos se desarrollarán más ampliamente durante el modelo de diseño.

A continuación describimos las clases entidad necesarias para cada caso de uso de acuerdo a la documentación generada durante el modelo de requisitos del capítulo anterior. Nótese que las clases son obtenidas del dominio del problema generado en el modelo de requisitos. Si fueran necesarias nuevas clases entidad habría que modificar el dominio del problema anterior.

?? *Validar Usuario*: Este caso de uso requiere validar información exclusivamente guardada en el registro de usuario, lo que se hace en la clase entidad *RegistroUsuario*, utilizada también por el caso de uso *RegistrarUsuario*. En la Figura 7.19 se muestran las clases entidad identificadas en este caso de uso.

RegistroUsuario

Figura 7.19 Clases entidad identificadas del caso uso *Validar Usuario*.

?? *Ofrecer Servicios*: Este caso de uso administra las opciones de servicio y no requiere de ninguna clase entidad.
?? *Registrar Usuario*: Este caso de uso requiere guardar información exclusivamente acerca del usuario, lo que se hace en la clase entidad *RegistroUsuario*. En la Figura 7.20 se muestran las clases entidad identificadas en este caso de uso.

RegistroUsuario

Figura 7.20 Clases entidad identificadas del caso uso *Registrar Usuario*.

?? *Registrar Tarjeta*: Este caso de uso requiere guardar información exclusivamente acerca de la tarjeta del usuario, lo que se hace en la clase entidad *RegistroTarjeta*. En la Figura 7.21 se muestran las clases entidad identificadas en este caso de uso.

RegistroTarjeta

Figura 7.21 Clases entidad identificadas del caso uso *Registrar Tarjeta*.

?? *Consultar Información*: Este caso de uso requiere de toda la información relacionada con consultas. Se pueden tomar las clases del dominio del problema y quitar aquellas relacionadas con registros y reservaciones. De tal

manera tenemos las clases entidad *Asiento*, *Avión*, *Tarifa*, *Aeropuerto*, *Aerolínea*, *Vuelo* y *Horario*. En la Figura 7.22 se muestran las clases entidad identificadas en este caso de uso.

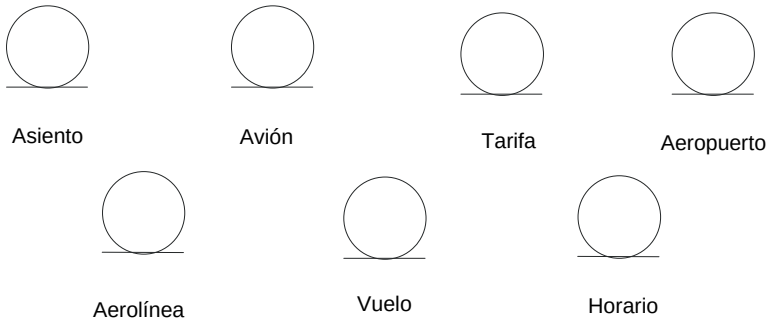

Figura 7.22 Clases entidad identificadas del caso uso *Consultar Información*.

?? *Hacer Reservación*: Este caso de uso requiere de toda la información relacionada con reservaciones. Se pueden tomar las clases del dominio del problema y quitar aquellas relacionadas con registros. De tal manera tenemos las clases entidad *Asiento*, *Avión*, *Tarifa*, *Aeropuerto*, *Aerolínea*, *Vuelo*, *Horario*, *ViajeroFrecuente*, *Pasajero* y *Reservación*. En la Figura 7.23 se muestran las clases entidad identificadas en este caso de uso.

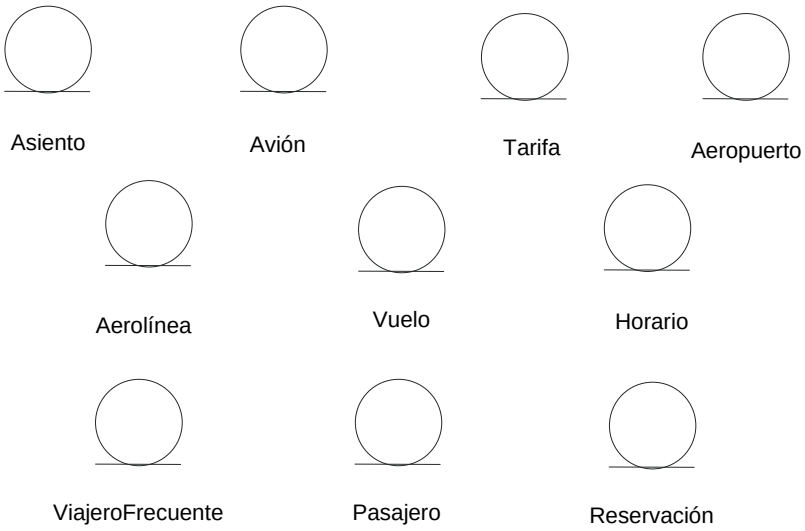

Figura 7.23 Clases entidad identificadas del caso uso *Hacer Reservación*.

?? *Pagar Reservación*: Este caso de uso requiere de toda la información relacionada con reservaciones. Se pueden tomar las clases del dominio del problema y quitar aquellas relacionadas con registro de usuario. En este caso de uso es necesario el registro de tarjeta para poder completar el pago o el reembolso. De tal manera tenemos las clases entidad *Asiento*, *Avión*, *Tarifa*, *Aeropuerto*, *Aerolínea*, *Vuelo*, *Horario*, *ViajeroFrecuente*, *Pasajero*, *Reservación* y *RegistroTarjeta*. En la Figura 7.24 se muestran las clases entidad identificadas en este caso de uso.

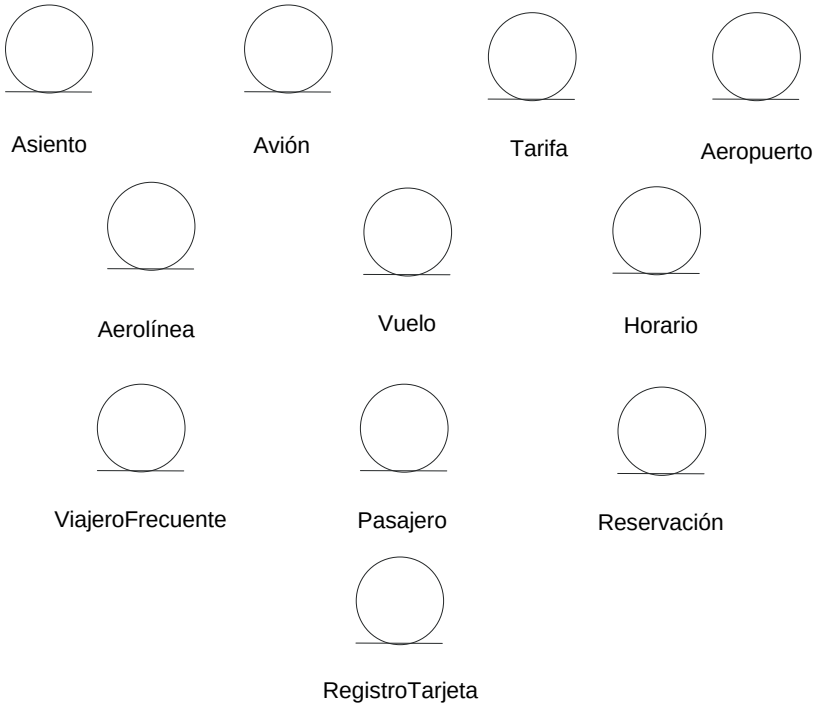

Figura 7.24 Clases entidad identificadas del caso uso *Pagar Reservación*.

En la Tabla 7.2 se muestran el resumen de los casos de uso identificados durante el modelo de requisitos junto con clases entidad correspondientes.

Caso de Uso	Clases Entidad
Validar Usuario	*RegistroUsuario*
Ofrecer Servicios	
Registrar Usuario	*RegistroUsuario*
Registrar Tarjeta	*RegistroTarjeta*
Consultar Información	*Asiento, Avión, Tarifa, Aeropuerto, Aerolínea, Vuelo, Horario*
Hacer Reservación	*Asiento, Avión, Tarifa, Aeropuerto, Aerolínea, Vuelo, Horario, ViajeroFrecuente, Pasajero, Reservación*
Pagar Reservación	*Asiento, Avión, Tarifa, Aeropuerto, Aerolínea, Vuelo, Horario, ViajeroFrecuente, Pasajero, Reservación, RegistroTarjeta*

Tabla 7.2 Relación entre casos de uso y clases entidad para el sistema de reservaciones de vuelo.

Control

Hasta ahora se han identificado partido objetos borde y entidad a partir de cada caso de uso. En algunas situaciones todo un caso de uso pudiera implementarse exclusivamente mediante estos dos tipos de objetos. De tal manera no se necesitaría ningún objeto control para el respectivo caso de uso. Sin embargo, en la mayoría de los casos de uso, existe un comportamiento que no se puede asignar de forma natural a ninguno de los otros dos tipos de objetos, ya que realmente no pertenece de manera natural a ninguno de ellos. Una posibilidad es repartir el comportamiento entre los dos tipos de objetos, como lo sugieren algunos métodos, pero la solución no es buena si se considera el aspecto de extensibilidad. Un cambio en el comportamiento podría afectar varios objetos dificultando la su modificación. Por lo tanto, para evitar estos problemas tal comportamiento se asigna en *objetos control*. Sin embargo, es difícil lograr un buen balance en la asignación del comportamiento entre los objetos entidad, borde y control. Los objetos de control típicamente actúan como “pegamento” entre los otros tipos de objetos y por lo tanto proveen la comunicación entre los demás tipos de objetos. Son típicamente los más efímeros de todos los tipos de objetos, dependiendo de la existencia del propio caso de uso.

Los objetos control se identifican directamente de los casos de uso. Como primera aproximación, se asigna un objeto control a cada caso de uso, concreto y abstracto. Dado que se asigna inicialmente el comportamiento a los objetos borde y entidad para cada caso de uso, el comportamiento restante se asigna a los objetos control. A menudo un

manera de asignar el comportamiento es modelar inicialmente el caso de uso sin ningún objeto control, o sea sólo utilizar objetos borde y objetos entidad. Cuando tal modelo se ha desarrollado, se verá que hay ciertos comportamientos que no se asignan de forma natural, ni en los objetos entidad ni en los objetos borde, o peor aún, se dispersan sobre varios objetos. Estos comportamientos deben ubicarse en los objetos control. Sin embargo, puede darse la situación donde no queda comportamiento restante para modelar en el caso de uso. En tal caso no se necesita un objeto control. Otra situación es si el comportamiento que queda, después de distribuir el comportamiento relevante entre objetos borde y entidad, es demasiado complicado, la funcionalidad puede ser dividida en varios objetos control. Por otro lado, si un caso de uso se acopla a varios actores esto puede indicar que existen variados comportamientos en relación a los diferentes actores y por lo tanto deben asignarse varios objetos control. La meta debe ser ligar solo un actor con cada objeto control ya que los cambios en los sistemas a menudo son originados por los actores y de tal manera se logra modularizar los posibles cambios.
La estrategia de asignación de control se debe decidir según cada aplicación. En la mayoría de los casos, sin embargo, se promueve la separación del control de un caso de uso en un objeto control que delega funcionalidad de manejo más local a los otros dos tipos de objetos.
En el sistema de reservaciones de vuelo asignaremos inicialmente un objeto control para cada caso como se verá a continuación. A estas clases las llamaremos manejadores o controladores para distinguir de los demás estereotipos.

?? *Validar Usuario*: Este caso de uso requiere un controlador para manejar la validación del registro de usuario. Dado que esto utiliza la misma información de registro podemos como enfoque inicial utilizar la misma clase control que en el caso de uso anterior, por lo cual utilizamo la clase control *ManejadorRegistroUsuario*. En la Figura 7.25 se muestra la clase control para este caso de uso.

Figura 7.25 Clase control para el caso uso *Validar Usuario*.

?? *Ofrecer Servicios*: Este caso de uso requiere un controlador para administrar los aspectos generales de los sevicios. Como enfoque inicial utilizar utilizaremos una clase general de control que llamaremos *ManejadorServicio*. En la Figura 7.26 se muestra la clase control para este caso de uso.

Figura 7.26 Clase control para el caso uso *Ofrecer Servicios*.

?? *Registrar Usuario*: Este caso de uso requiere de un controlador para manejar la información, lo que haremos mediante la clase control *ManejadorRegistroUsuario*. En la Figura 7.27 se muestra la clase control para este caso de uso.

Figura 7.27 Clase control para el caso uso *Registrar Usuario*.

?? *Registrar Tarjeta*: Este caso de uso requiere una clase controladora para administrar el registro de la tarjeta del usuario, lo que haremos mediante la clase control *ManejadorRegistroTarjeta*. En la Figura 7.28 se muestra la clase control para este caso de uso.

Figura 7.28 Clase control para el caso uso *Registrar Tarjeta*.

?? *Consultar Información*: Este caso de uso requiere de un controlador para manejar la información y las bordes relacionadas con las consultas, lo que se hace en la clase control *ManejadorConsultas*. Dado que se tiene tres tipos de consultas distintas, podemos incluir tres controladores especializados, *ManejadorConsultaHorarios*,

ManejadorConsultaTarifas y *ManejadorConsultaEstado*, para las consultas respectivas. En la Figura 7.29 se muestran las clases control identificadas en este caso de uso.

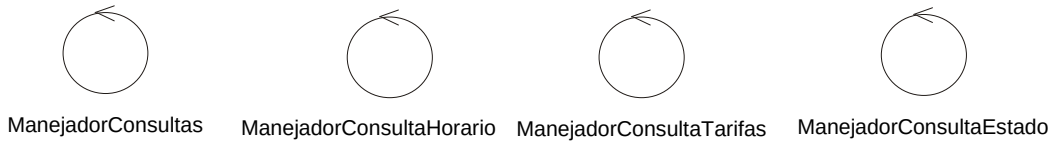

Figura 7.29 Clases control para el caso uso *Consultar Información*.

?? *Hacer Reservación*: Este caso de uso requiere de un controlador para manejar lo relacionado con las reservas, lo que se hace mediante la clase control *ManejadorReservas*. En la Figura 7.30 se muestra la clase control para este caso de uso.

Figura 7.30 Clase control para el caso uso *Hacer Reservación*.

?? *Pagar Reservación*: Este caso de uso requiere administrar lo relacionado con los pagos, lo que se haremos mediante la clase control *ManejadorPagos*. En la Figura 7.31 se muestran la clase control para en este caso de uso.

Figura 7.31 Clase control para el caso uso *Pagar Reservación*.

Adicionalmente, por lo general es siempre bueno incluir un controlador principal para administrar los aspectos generales del sistema, incluyendo la pantalla principal. A esta clase la llamaremos *ManejadorPrincipal*, la cual se muestra en la Figura 7.32.

Figura 7.32 Clase control para el sistema completo.

En la Tabla 7.3 se muestran el resumen de los casos de uso identificados durante el modelo de requisitos junto con clases control correspondientes.

Caso de Uso	**Clases Control**
Validar Usuario	*ManejadorRegistroUsuario*
Ofrecer Servicios	*ManejadorServicio*
Registrar Usuario	*ManejadorRegistroUsuario*
Registrar Tarjeta	*ManejadorRegistroTarjeta*
Consultar Información	*ManejadorConsultas, ManejadorConsultaHorarios, ManejadorConsultaTarifas, ManejadorConsultaEstado*
Hacer Reservación	*ManejadorReservas*
Pagar Reservación	*ManejadorPagos*
	ManejadorPrincipal

Tabla 7.3 Relación entre casos de uso y clases control para el sistema de reservaciones de vuelo.

7.3 Clases según Casos de Uso

En esta sección mostramos un diagrama de clases para cada caso de uso de acuerdo a las clases identificadas en la sección anterior. En estos diagramas incluiremos de manera preliminar asociaciones y multiplicidad. Para simplificar este proceso de asignación de asociaciones y para ser consistentes entre diagramas, asignaremos a la clase control de cada caso de uso como el “centro de las comunicaciones” para todas las clases borde y entidad

pertenecientes al mismo caso de uso. Estas clases borde y entidad no estarán asociadas entre si por el momento. Asimismo asociaremos entre si a las clases control utilizadas en el caso de uso. Y finalmente se asociarán todas las clases borde correspondiente a pantallas con la clase *InterfaceUsuario*. Nótese que sólo se incluyen las clases borde correspondientes a pantallas que se consideren "esenciales" para el caso de uso, en otras palabras aquellas que implementan los flujos principales pero no tanto ligas adicionales de navegación entre pantallas.
En relación a la multiplicidad, asignaremos todas las asociaciones como de "uno-uno" con excepción de las relaciones entre clases entidades que se mantendrán de acuerdo al dominio del problema y entre clases control y clases entidad donde por lo general será "uno-muchos". La razón para esto es que típicamente sólo se requiere una instancia de cada clase control e borde para implementar el caso de uso, mientras que las clases entidad se instancian múltiples veces ya que corresponden a diversas instancias de información, como por ejemplo, múltiples usuarios. Como veremos en la sección de diagramas de interacción más adelante en el capítulo, a partir de estas relaciones estamos forjando la lógica de control del caso de uso. En las siguientes secciones veremos estos casos con mayor detalle.

Validar Usuario

El caso de uso *Validar Usuario* involucra una clase control *ManejadorRegistroUsuario* que es encargada de controlar la información de *RegistroUsuario* y las clases borde *InterfaceUsuario* e *InterfaceBaseDatosRegistro*. Agregamos también la clase *PantallaPrincipal* por recibir la información de registro a ser validada y al *ManejadorPrincipal* por ser el controlador de la pantalla anterior. En la Figura 7.33 se muestran las clases identificadas en este caso de uso.

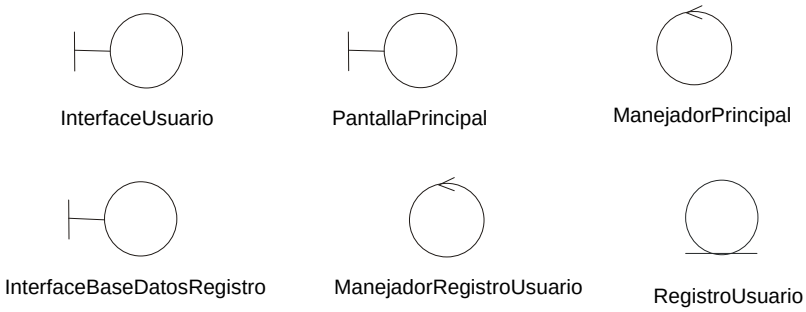

Figura 7.33 Clases identificadas para el caso uso *Validar Usuario*.

Ofrecer Servicios

El caso de uso *Ofrecer Servicios* involucra una clase control *ManejadorServicio* que es encargada de controlar la pantalla *PantallaServicio*. Agregamos también la clase borde *InterfaceUsuario*. En la Figura 7.34 se muestran las clases identificadas en este caso de uso.

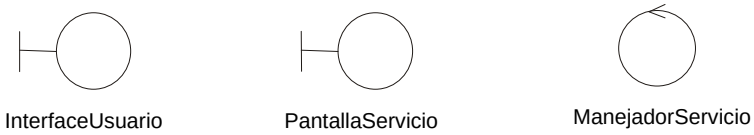

Figura 7.34 Clases identificadas para el caso uso *Ofrecer Servicios*.

Registrar Usuario

El caso de uso *Registrar Usuario* involucra una clase control *ManejadorRegistroUsuario* que es encargada de controlar la información de *RegistroUsuario* y las clases borde correspondiente a las pantallas *PantallaCrearRegUsuario* y *PantallaObtenerRegistroUsuario*, además de las clases borde *InterfaceUsuario* e *InterfaceBaseDatosRegistro*. En la Figura 7.35 se muestran las clases identificadas en este caso de uso.

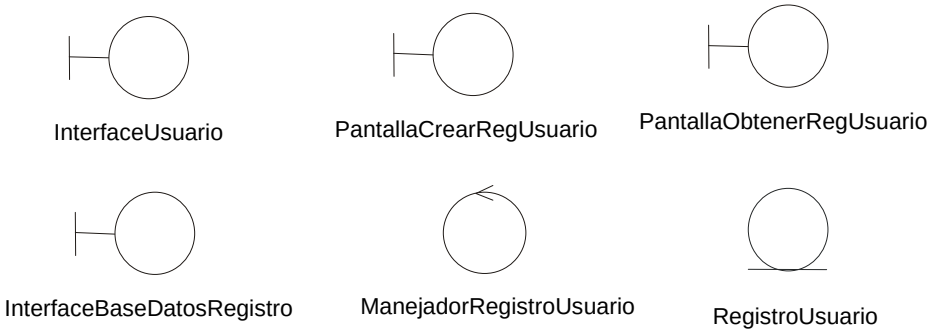

Figura 7.35 Clases identificadas para el caso uso *Registrar Usuario*.

Registrar Tarjeta

El caso de uso *Registrar Tarjeta* involucra una clase control *ManejadorRegistroTarjeta* que es encargada de controlar la información de *RegistroTarjeta* y las clases borde correspondiente a las pantallas *PantallaCrearRegTarjeta* y *PantallaObtenerRegistroTarjeta* además de las clases borde *InterfaceUsuario* e *InterfaceBaseDatosRegistro*. En la Figura 7.36 se muestran las clases identificadas en este caso de uso.

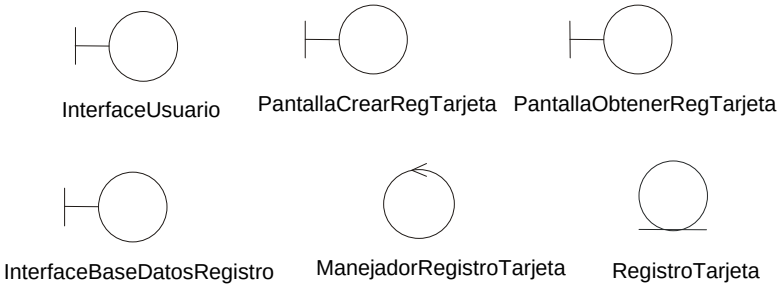

Figura 7.36 Clases identificadas para el caso uso *Registrar Tarjeta*.

Consultar Información

El caso de uso *Consultar Información* involucra una clase control *ManejadorConsultas* que es encargada de controlar toda los diferentes tipos de consultas junto con la clase borde correspondiente a la pantalla *PantallaConsultas*, además de las clases borde *InterfaceUsuario* e *InterfaceBaseDatosRegistro*. Dado que este caso de uso tiene tres subflujos importantes, en lugar de describirlos en un sólo diagrama, lo haremos en tres diagramas separados como veremos más adelante. En la Figura 7.37 se muestran las clases principales identificadas en este caso de uso.

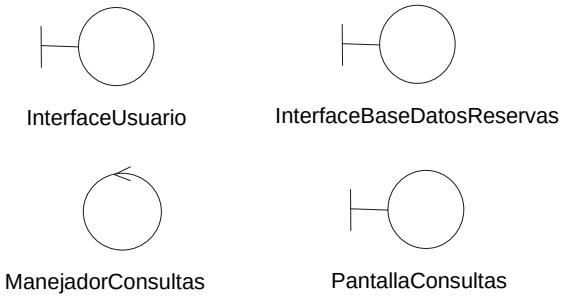

Figura 7.37 Clases identificadas para el caso uso *Consultar Información*.

Consultar Horarios

El subflujo *Consultar Horarios* del caso de uso *Consultar Información* involucra a todas las clases del diagrama de la Figura 7.37, las cuales no volvemos a incluir en el diagrama. Se incluyen en el nuevo diagrama las clases borde correspondiente a las pantallas *PantallaConsultaHorarios* y *PantallaResultadoHorarios* además de las clases entidad *Vuelo*, *Aeropuerto*, *Horario* y *Aerolínea* junto con la clase control *ManejadorConsultaHorarios*. El resto de

las clases entidad del dominio del problema no son necesarias para este subflujo. En la Figura 7.38 se muestran las clases identificadas en este subflujo.

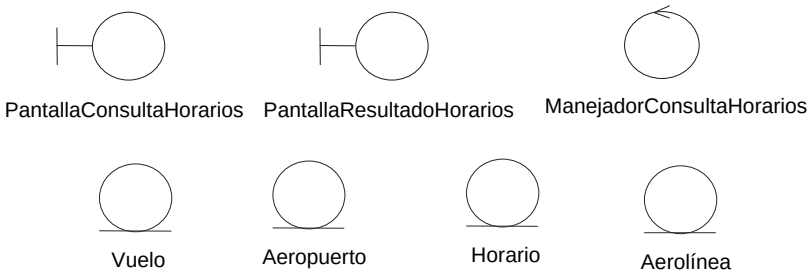

Figura 7.38 Clases identificadas para el subflujo *Consultar Horarios* del caso uso *Consultar Información*.

Consultar Tarifas

El subflujo *Consultar Tarifas* del caso de uso *Consultar Información* involucra a todas las clases del diagrama de la Figura 7.37, las cuales no volvemos a incluir en el diagrama. Se incluyen en el nuevo diagrama las clases borde correspondiente a las pantallas *PantallaConsultaTarifas* y *PantallaResultadoTarifas* además de las clases entidad *Vuelo*, *Aeropuerto*, *Horario*, *Aerolínea* y *Tarifa* junto con la clase control *ManejadorConsultaTarifas*. El resto de las clases entidad del dominio del problema no son necesarias para este subflujo. En la Figura 7.39 se muestran las clases identificadas en este caso de uso.

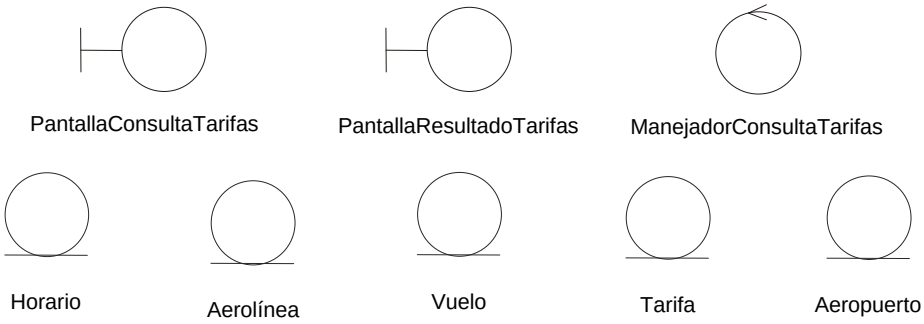

Figura 7.39 Clases identificadas para el subflujo *Consultar Tarifas* del caso uso *Consultar Información*.

Consultar Estado

El subflujo *Consultar Estado* del caso de uso *Consultar Información* involucra a todas las clases del diagrama de la Figura 7.37, las cuales no volvemos a incluir en el diagrama. Se incluyen en el nuevo diagrama las clases borde correspondiente a las pantallas *PantallaConsultaEstado* y *PantallaResultadoEstado* además de las clases entidad *Vuelo*, *Aeropuerto*, *Horario*, *Aerolínea* y *Avión* junto con la clase control *ManejadorConsultaEstado*. El resto de las clases entidad del dominio del problema no son necesarias para este subflujo. En la Figura 7.40 se muestran las clases identificadas en este caso de uso.

Figura 7.40 Clases identificadas para el subflujo *Consultar Estado* del caso uso *Consultar Información*.

Hacer Reservación

El caso de uso *Hacer Reservación* involucra una clase control *ManejadorReservas* que es encargada de controlar las reservaciones junto con las clases bordes correspondiente a la pantalla *PantallaClaveReservas*, *PantallaCrearReservaVuelos* y *PantallaRecordReservaVuelos* además de las clases borde *InterfaceUsuario* e *InterfaceBaseDatosReserva*. Se incluyen en el diagrama todas las clases entidad necesarias, que son *Vuelo*, *Asiento*, *Avión*, *Tarifa*, *Horario*, *Aerolínea*, *Aeropuerto*, *Reservación*, *Pasajero* y *ViajeroFrecuente*. En la Figura 7.41 se muestran las clases identificadas en este caso de uso.

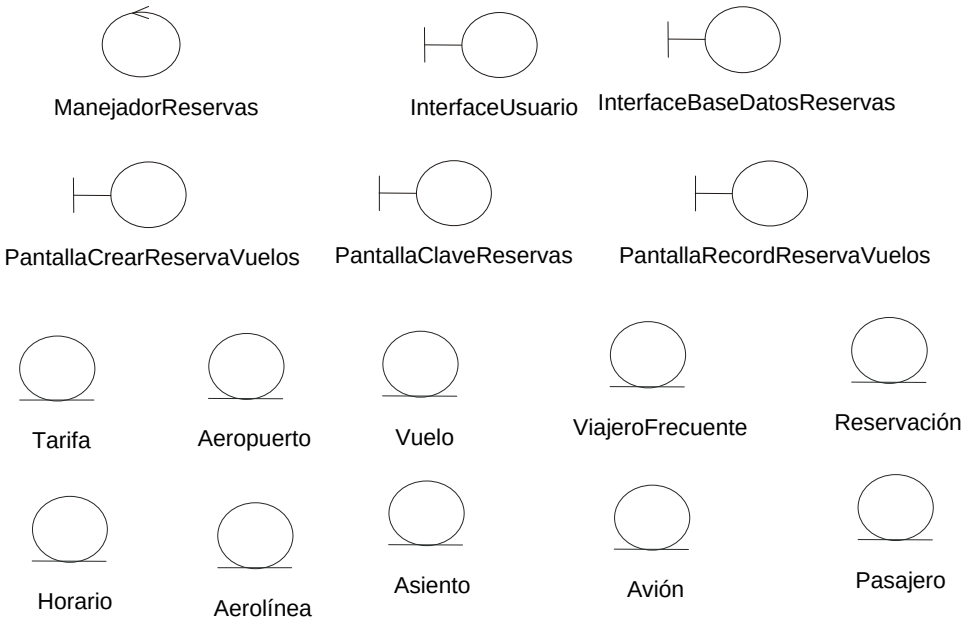

Figura 7.41 Clases identificadas para el caso uso *Hacer Reservación*.

Pagar Reservación

El caso de uso *Pagar Reservación* involucra una clase control *ManejadorPagos* que es encargada de controlar la información de pagos y las clases borde correspondiente a las pantallas *PantallaPagarRegTarjeta* y *PantallaReembolsarRegTarjeta* además de las clases borde *InterfaceUsuario* e *InterfaceBaseDatosReserva*.
Se incluye también la clase *RegistroTarjeta* dado que es necesaria para obtener la información de la tarjeta de crédito. En la Figura 7.42 se muestran las clases identificadas en este caso de uso.

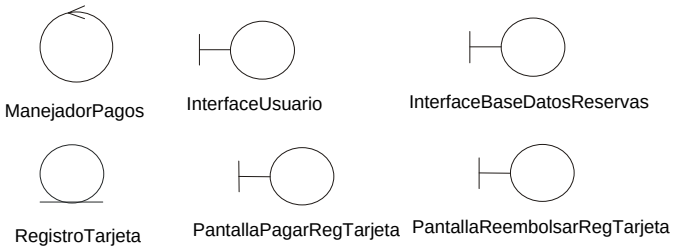

Figura 7.42 Clases identificadas para el caso uso *Pagar Reservación*.

7.4 Diagramas de Secuencias

Una vez identificadas las clases anteriores, proseguimos describiendo los casos de uso del modelo de requisitos según la lógica que deberán presentar estas clases para lograr la funcionalidad descrita en los diversos casos de uso. Este es un paso muy importante ya que en base a la lógica propuesta para esta descripción, definiremos la arquitectura de análisis tanto estructural como funcional.
Dada la complejidad y la importancia de estas descripciones, es importante probar las secuencias funcionales de los casos de uso flujos revisar qué tan bien nuestra lógica y arquitectura de clase resuelve la funcionalidad establecida. Para eso introducimos el concepto de *diagramas de secuencias*, *interacción* o *eventos*, los cuales describen como los

diferentes casos de uso son implementados mediante los objetos de nuestra arquitectura recién generados. Los diagramas correspondientes muestran la interacción entre los objetos participantes a nivel de eventos que se envían entre si, excluyendo cualquier detalle interno de ellos. El formato de un diagrama de secuencia se muestra en la Figura 7.43. Nótese que es un diagrama exclusivamente de objetos y no de clases. Sin embargo, los objetos son instancias de clases que al no tener ningún nombre particular se muestran a través del nombre de la clase subrayada y con el prefijo de ":", correspondiente a la notación para diagramas de objetos como se vio en el Capítulo 4.

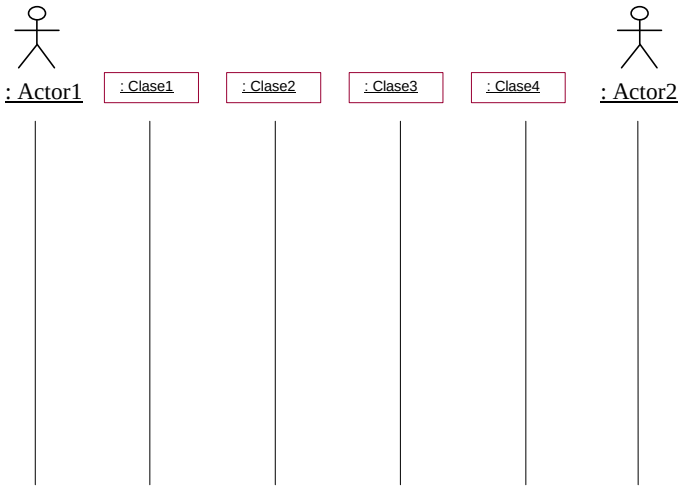

Figura 7.43 Diagrama de secuencia.

Cada clase participante se representa por una barra, dibujadas como líneas verticales en el diagrama. El orden entre las barras es insignificante y debe elegirse para dar la mayor claridad. Si hubieran varias instancias de las clases que interactúan entre si, se dibujarían las instancias como barras diferentes. Además de los objetos, es importante representar entidades externas al sistema en los diagramas de secuencia. De lo contrario este tipo de diagrama no sería demasiado útil en el modelo de análisis. En nuestro caso, estas entidades externas que se incluyen como barras adicionales en el diagrama representando *instancias* de los actores.

El eje de tiempo en el diagrama de secuencia es vertical (paralelo a las barras) y avanzando hacia abajo. El comienzo del diagrama de secuencia corresponde al inicio del caso de uso. El avance en el tiempo en el diagrama es controlado por los eventos, mientras que la distancia entre dos eventos en el diagrama no tiene relación con el tiempo real entre estos eventos. Más aún el tiempo no es necesariamente lineal en el diagrama, pudiendo existir concurrencia. El diagrama de secuencia de la Figura 7.44 muestra un ejemplo de estos eventos.

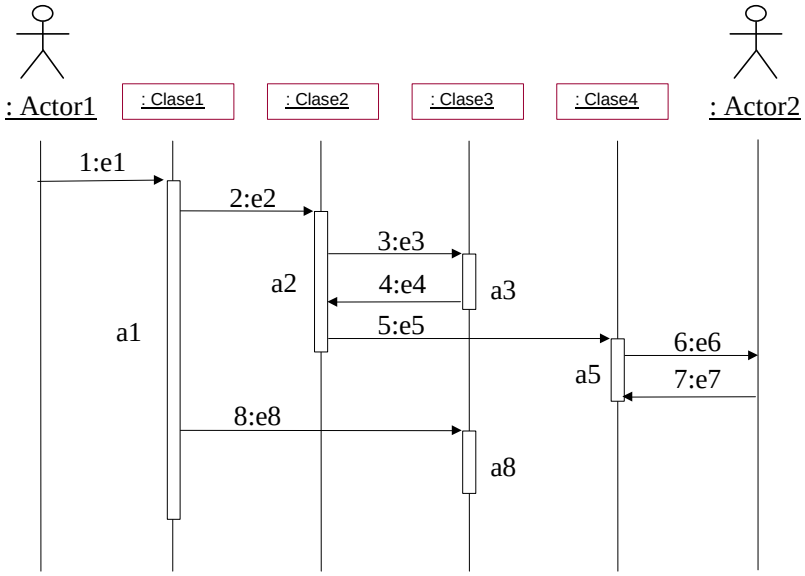

Figura 7.44 Diagrama de secuencia con eventos.

En el diagrama de la figura 7.44, las barras gruesas corresponden a *actividades* dentro del objeto, denominadas por *a*, mientras que las flechas corresponden a *eventos*, denominadas por *e*. Un evento se dibuja como una flecha horizontal que comienza en la barra correspondiente al objeto que lo envía y termina en la barra correspondiente al objeto que lo recibe. En este ejemplo los eventos son numerados sucesivamente utilizando el formato "*n*:" . Las actividades son iniciadas por el arribo de eventos y el tiempo que duran es sólo relevante en relación a eventos originados durante esa actividad, como en el caso de el objeto de la *Clase1* que recibe un evento *e1*, el cual inicia una actividad que consiste en primero generar un evento *e2* y posteriormente otro evento *e6*. Un aspecto importante que los diagramas de secuencia deben considerar es que las secuencias de eventos sean continuas y no allá interrupciones. Por otro lado la gran mayoría de los diagramas de secuencia, y por lo tanto los casos de uso, deben comenzar con un evento externo que se genera a partir de una de las instancias de los actores del sistema. Por ejemplo, consideremos la secuencia que comienza con el *Actor1* a través del evento *e1*. Este evento da lugar al evento *e2* el cual a su vez da lugar al evento *e3* y así sucesivamente hasta llegar el evento *e7*. Sin embargo se interrumpe la continuidad entre el evento *e7* y el evento *e8*. Esta situación es muy delicada y peligrosa ya que al no haber una dependencia directa entre ambos eventos y si consideramos que no existe una relación de tiempo real entre ningún evento, pues *e8* pudiera generarse en cualquier momento lo cual pudiera ocasionar inconsistencias en la aplicación. Esto se debe evitar a toda costa, cada nuevo evento debe generarse a partir de uno que se recibe con la única excepción de eventos iniciales que son generados por el sistema o típicamente por un actor. Nótese que los actores principales son los que típicamente deciden que casos de uso serán instanciados, a diferencia de los propios objetos y casos de uso secundarios que más bien responden a eventos que son recibidos.
Durante el modelo de análisis y como veremos en las siguientes secciones, utilizaremos los diagramas de secuencia para describir los *flujos principales* y *subflujos* para cada caso de uso. Esto ayudará a revisar si nuestra lógica es correcta y consistente al relacionar las casos de uso del modelo de requisitos con la arquitectura de clases del modelo de análisis.
Dado que existen múltiples posibles flujos de secuencia, nos concentraremos únicamente en los principales que definan flujos completos y que incluyen subflujos intermedios, incluyendo casos de uso de inclusión. Por lo tanto, describiremos los casos de uso de mayor interés. Nótese que los incicios de muchos de estos casos se repiten, lo cual incluiremos en los diversos casos de uso para no perder el flujo de eventos en el desarrollo de las diversas secuencias.

Registrar Usuario

En el caso de uso *Registrar Usuario* existen diversas secuencias que pueden ser instanciados por un usuario. En esta sección mostramos las secuencias que pudieran desarrollarse entre los objetos para asegurar que la lógica que estamos utilizando sea correcta y que corresponda a los casos de uso especificados en el modelo de requisitos. En particular mostraremos diagramas de secuencias para *Crear Registro Usuario*, *Actualizar Registro Usuario* y *Eliminar Registro Usuario*. Nótese que estas secuencias incluirán obviamente a los subflujos *Crear Registro Usuario* (*S*-1), *Actualizar Registro Usuario* (*S*-4) y *Eliminar Registro Usuario* (*S*-5), de manera correspondiente, junto con los casos de uso, *Validar Usuario* y *Ofrecer Servicios*, además de los los subflujos *Obtener Registro Usuario* (*S*-2) y *Administrar Registro Usuario* (*S*-3).

Crear Registro Usuario

El diagrama de secuencia para *Crear Registro Usuario* se muestra en el diagrama 7.45. Esta secuencia inicia en el flujo principal de *Registrar Usuario* que deberá incluir ciertas opciones definidas en *Validar Usuario* seguidos por el subflujo *Crear Registro Usuario* (*S*-1) y *Administrar Registro Usuario* (*S*-3).

?? *Validar Usuario*. Aunque la secuencia se inicia en el flujo princial de *Registrar Usuario*, se continúa inmediatamente con la inserción del caso de uso *Validar Usuario*, donde podemos iniciar con el *ManejadorPrincipal* solicitando el desplegado de la *PantallaPrincipal* mediante el evento *desplegarPantallaPrincipal*. Para continuar este secuencia, el usuario deberá seleccionar la opción de "*Registrar Por Primera Vez*" oprimiendo el botón correspondiente en la pantalla. La *InterfaceUsuario* por ser la controladora de las interfaces de usuario, recibe el evento y lo envía como un nuevo evento "*Registrar Por Primera Vez*" al *ManejadorPrincipal*. El *ManejadorPrincipal* que es el encargado de controlar la lógica general del sistema, reconoce que este evento corresponde a una actividad de registro y se lo envía como *crearRegUsuario* al *ManejadorRegistroUsuario*.

?? *Registrar Usuario* subflujo *Crear Registro Usuario* (*S*-1). En este momento el *ManejadorRegistroUsuario* reconoce el tipo de evento particular y solicita a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaCrearRegUsuario*. La *InterfaceUsuario* despliega esta pantalla,

algo que no se muestra en el diagrama por ser un evento interno. Para continuar con la lógica principal de este subflujo, el usuario debe llenar sus datos, que no se muestran aquí, y oprime el botón "*Registrar*" para que esta información sea enviada a la clase *InterfaceUsuario*. Es importante resaltar que los datos como tales no generan eventos ni son de importancia en estos diagramas. Lo que genera eventos son los botones en las pantallas. Siguiendo con nuestra lógica, la *InterfaceUsuario* envía el evento "*Registrar*" al *ManejadorRegistroUsuario*. Este controlador es responsable de guardar la información de registro del usuario, por lo cual envía el evento *crearRegUsuario* a la *InterfaceBaseDatosRegistro*. Nótese que como en el caso de los datos, los objetos entidad como la clase *RegistroUsuario*, tampoco son mostrados en el diagrama, dado que no agregan eventos interesantes para la lógica del sistema. Incluso se omiten del diagrama todas las clases correspondientes a pantallas ya que sus eventos importantes son manejados por la *InterfaceUsuario*. Prosiguiendo con nuestra lógica, la *InterfaceBaseDatosRegistro* envía el evento *crearRegUsuario* al actor *Base de Datos Registro*s. Este actor debe responder de alguna manera, y lo hace mediante un *OK* el cual es luego enviado de manera sucesiva hasta llegar al *ManejadorRegistroUsuario*.

?? *Registrar Usuario* subflujo *Administrar Registro Usuario* (*S*-3). A continuación pasamos al subflujo *Administrar Registro Usuario* (*S*-3) donde el El *ManejadorRegistroUsuario* envía el evento *desplegarPantallaObtenerRegUsuario* a la *InterfaceUsuario*. En ese momento el *Usuario* presiona *Salir*, dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido de los subflujos *Crear Registro Usuario* (*S*-1) y *Administrar Registro Usuario* (*S*-3) del caso de uso *Registrar Usuario*.

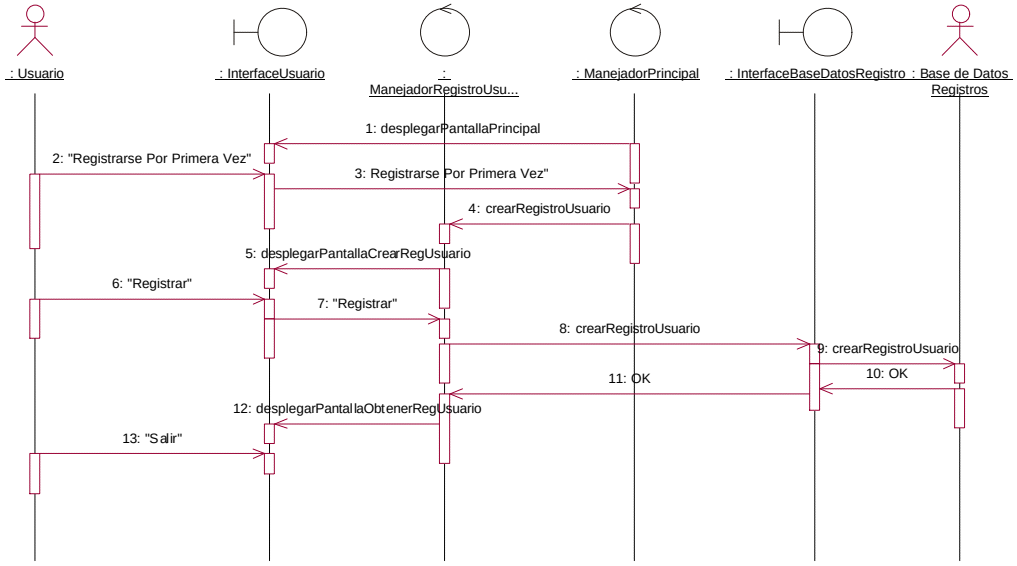

Figura 7.45 Diagrama de secuencia *Crear Registro Usuario* del caso de uso *Registrar Usuario*.

Vale la pena resaltar dos aspectos importantes en el diagrama. El primer aspecto es que en ningún momento se corta el flujo de los eventos. La única excepción en el diagrama son los eventos generados por el *Usuario* que sólo se pueden generar a partir de que las pantallas hayan sido desplegadas por lo cual realmente no hay ninguna interrupción lógica, sino más bien que no se muestran flujos de evento entre la *InterfaceUsuario* y el *Usuario* ya que estos son todos visuales (tendría el usuario que tener por ejemplo algún botón en su cuerpo para que realmente se mostrará tal evento). El segundo aspecto es que los eventos entre objetos son como una cascada donde un objeto le envía un evento a otro y este otro objeto le devuelve posteriormente un evento en respuesta, de manera directa o indirecta. Por ejemplo, el evento "7" es en respuesta al evento "5", mientras que el evento "11" es en respuesta al evento "8". Un ejemplo de respuesta indirecta es el caso del evento "3" respondido mediante los eventos "4" y "5". Una situación particular se da con el último evento correspondiente a "*Salir*", el cual interrumpe estos flujos de respuesta ya que en algún lugar se debe terminar la secuencia.

Actualizar Registro Usuario

El diagrama de secuencia *Actualizar Registro Usuario* se muestra en el diagrama 7.46. Esta secuencia inicia en el flujo principal de *Registrar Usuario* que incluye las opciones de validación definidas en *Validar Usuario*, seguidos por parte de *Ofrecer Servicios*, para luego proseguir con el subflujo *Obtener Registro Usuario* (*S*-2), *Administrar Registro Usuario* (*S*-3) y obviamente *Actualizar Registro Usuario* (*S*-4), para completar la secuencia de la actualización.

?? *Validar Usuario*. Nuevamente, aunque la secuencia se inicia en el flujo princial de *Registrar Usuario*, se continúa inmediatamente con la inserción del caso de uso *Validar Usuario*, donde podemos iniciar con el *ManejadorPrincipal* enviando el evento *desplegarPantallaPrincipal* a la *InterfaceUsuario*, el cual despliega la *PantallaPrincipal*. En este momento el usuario debe validarse, insertando su login y contraseña. Al ser datos, estos no generan ningún evento. Una vez el usuario genere el evento "*OK*" oprimiendo el botón correspondiente, se instancia la validación. La *InterfaceUsuario* envía el mismo evento al *ManejadorPrincipal*. El *ManejadorPrincipal* reconoce que este evento corresponde a una actividad de registro y envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. Este controlador reconoce el tipo de evento particular y solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante un evento adicional con el mismo nombre. La *InterfaceBaseDatosRegistro* envía a su vez un evento similar al actor *Base de Datos Registros*, el cual contesta con un *OK* si la validación es buena. Dado que sólo consideramos una secuencia de eventos, una validación incorrecta se mostraría en otro diagrama. El *OK* es sucesivamente enviado a la *InterfaceBaseDatosRegistro*, de allí a *ManejadorRegistroUsuario* y luego a *ManejadorPrincipal* como respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios*.

?? *Ofrecer Servicios*. A continuación podemos pasar al caso de uso *Ofrecer Servicios* donde el *ManejadorServicio* solicita entonces a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "*Obtener Registro*". Este evento es enviado de la *InterfaceUsuario* al *ManejadorServicio*. El *ManejadorServicio* envía el evento *obtenerRegistroUsuario* al *ManejadorRegistroUsuario*.

?? *Registrar Usuario* subflujo *Obtener Registro Usuario* (*S*-2). A continuación regresamos al caso de uso *Registrar Usuario* subflujo *Obtener Registro Usuario* (*S*-2) donde el *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que obtenga el registro correspondiente mediante *obtenerRegistroUsuario*. Nuevamente, la *InterfaceBaseDatosRegistro* le pasa un evento similar al actor *Base de Datos Registros*, el cual contesta con un *OK*. Junto a este *OK* deben enviarse los propios datos, los cuales no son mostrados en el diagrama de secuencia por no generar eventos. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorRegistroUsuario*.

?? *Registrar Usuario* subflujo *Administrar Registro Usuario* (*S*-3). A continuación podemos pasar al subflujo *Administrar Registro Usuario* (*S*-3) donde el *ManejadorRegistroUsuario* solicita a la *InterfaceUsuario* desplegar la pantalla de información de registro mediante el evento *desplegarPantallaObtenerRegUsuario*. En este momento el usuario actualiza sus datos, que no se muestran aquí y oprime el botón "*Actualizar*".

?? *Registrar Usuario* subflujo *Actualizar Registro Usuario* (*S*-4). Siguiendo con la lógica, la *InterfaceUsuario* envía el mismo evento al *ManejadorRegistroUsuario*, el cual es responsable de actualizar el registro, por lo cual envía el evento *actualizarRegistroUsuario* a la *InterfaceBaseDatosRegistro*. La *InterfaceBaseDatosRegistro* envía el evento *actualizarRegistroUsuario* al actor *Base de Datos Registro*s. Este actor responde mediante un *OK* el cual es luego enviado de manera sucesiva hasta llegar al *ManejadorRegistroUsuario*.

?? *Registrar Usuario* subflujo *Administrar Registro Usuario* (*S*-3). A continuación podemos pasar al subflujo *Administrar Registro Usuario* (*S*-3) donde el *ManejadorRegistroUsuario* envía el evento *desplegarPantallaObtenerRegUsuario* a la *InterfaceUsuario*. En ese momento el *Usuario* presiona "*Salir*", dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios*, para luego continuar en el caso de uso *Registrar Usuario* con los subflujos *Obtener Registro Usuario* (*S*-2), *Administrar Registro Usuario* (*S*-3) y *Actualizar Registro Usuario* (*S*-4).

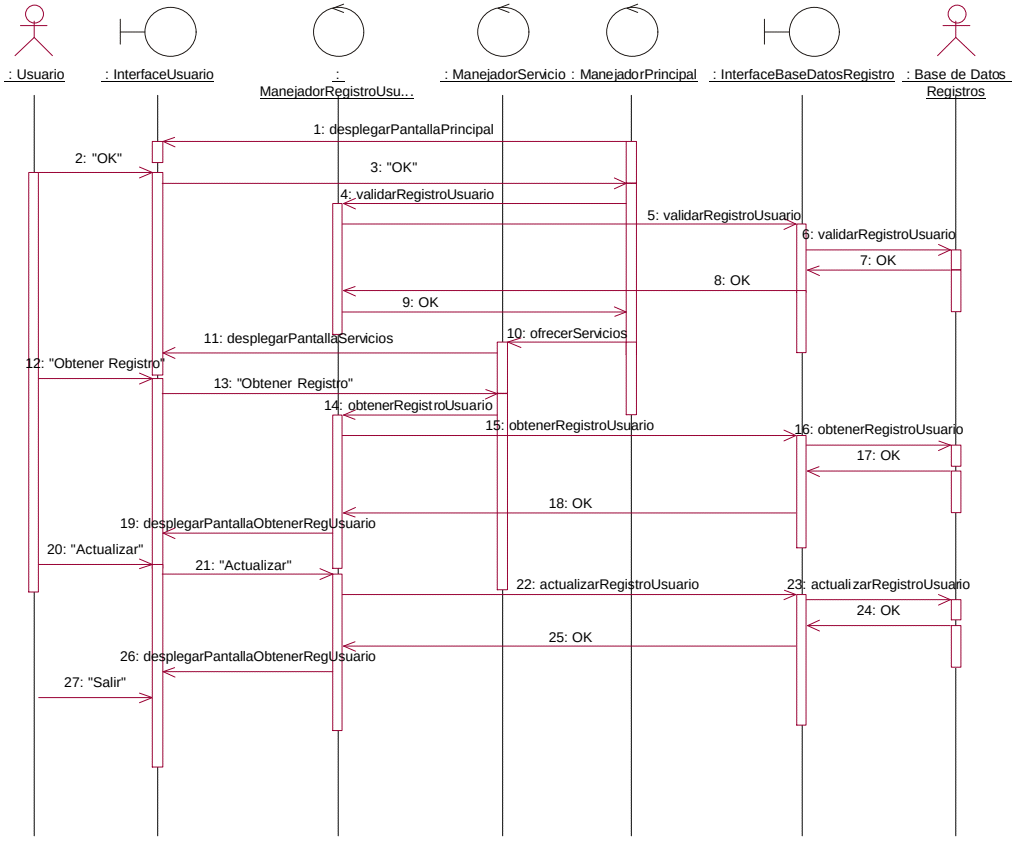

Figura 7.46 Diagrama de secuencia *Actualizar Registro Usuario* del caso de uso *Registrar Usuario*.

Eliminar Registro Usuario

El diagrama de secuencia para el subflujo *Eliminar Registro Usuario* se muestra en el diagrama 7.47. Esta secuencia es bastante similar a la de *Actualizar Registro Usuario* ya que inicia en el flujo principal de *Registrar Usuario* que incluye las opciones de validación definidas en *Validar Usuario*, seguidos por ciertos aspectos de *Ofrecer Servicios*, para luego proseguir con el subflujo *Obtener Registro Usuario* (*S*-2), *Administrar Registro Usuario* (*S*-3) y obviamente *Eliminar Registro Usuario* (*S*-5), para completar la secuencia de la eliminación. Nótese que solamente cambia el último subflujo.

?? *Validar Usuario*. Nuevamente, se comienza en *Validar Usuario* con la secuencia con el evento *desplegarPantallaPrincipal* de la clase *ManejadorPrincipal* al *InterfaceUsuario*. El Usuario se valida enviando el evento "*OK*". La *InterfaceUsuario* envía el evento "*OK*" al *ManejadorPrincipal*. El *ManejadorPrincipal* envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. El *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante el mismo evento. La *InterfaceBaseDatosRegistro* envía a su vez el evento a la *Base de Datos Registros*, el cual contesta con un *OK* si la validación es buena. El *OK* es enviado a la *InterfaceBaseDatosRegistro*, de allí a *ManejadorRegistroUsuario* y luego a *ManejadorPrincipal* como respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios*.

?? *Ofrecer Servicios*. A continuación podemos pasar al caso de uso *Ofrecer Servicios* donde el *ManejadorServicio* solicita entonces a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "*Obtener Registro*". Este evento es enviado de la *InterfaceUsuario* al *ManejadorServicio*. El *ManejadorServicio* envía el evento *obtenerRegistroUsuario* al *ManejadorRegistroUsuario.*

?? *Registrar Usuario* subflujo *Obtener Registro Usuario* (*S*-2). A continuación regresamos al caso de uso *Registrar Usuario* subflujo *Obtener Registro Usuario* (*S*-2) donde el *ManejadorRegistroUsuario* solicita a la

InterfaceBaseDatosRegistro que obtenga el registro correspondiente mediante *obtenerRegistroUsuario*. Nuevamente, la *InterfaceBaseDatosRegistro* le pasa un evento similar al actor *Base de Datos Registros*, el cual contesta con un *OK*. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorRegistroUsuario*.

?? *Registrar Usuario* subflujo *Administrar Registro Usuario* (*S*-3). A continuación podemos pasar al subflujo *Administrar Registro Usuario* (*S*-3) donde el *ManejadorRegistroUsuario* solicita a la *InterfaceUsuario* desplegar la pantalla de información de registro mediante el evento *desplegarPantallaObtenerRegUsuario*. Hasta aquí la secuencia es similar a *Actualizar Registro Usuario*. En este momento el usuario oprime el botón "*Eliminar*".

?? *Registrar Usuario* subflujo *Eliminar Registro Usuario* (*S*-3). Siguiendo con la lógica, La *InterfaceUsuario* envía el mismo evento al *ManejadorRegistroUsuario*, el cual es responsable de eliminar el registro, por lo cual envía el evento *eliminarRegistroUsuario* a la *InterfaceBaseDatosRegistro*. La *InterfaceBaseDatosRegistro* envía el evento *eliminarRegistroUsuario* al actor *Base de Datos Registro*s. Este actor responde mediante un *OK* el cual es luego enviado de manera sucesiva hasta llegar al *ManejadorRegistroUsuario*.

?? *Registrar Usuario* subflujo *Administrar Registro Usuario* (*S*-3). A continuación podemos pasar al subflujo *Administrar Registro Usuario* (*S*-3) donde el *ManejadorRegistroUsuario* envía el evento *desplegarPantallaCreaeRegUsuario* a la *InterfaceUsuario*. En ese momento el *Usuario* presiona "*Salir*", dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios*, para luego continuar en el caso de uso *Registrar Usuario* con los subflujos *Obtener Registro Usuario* (*S*-2), *Administrar Registro Usuario* (*S*-3) y *Eliminar Registro Usuario* (*S*-4).

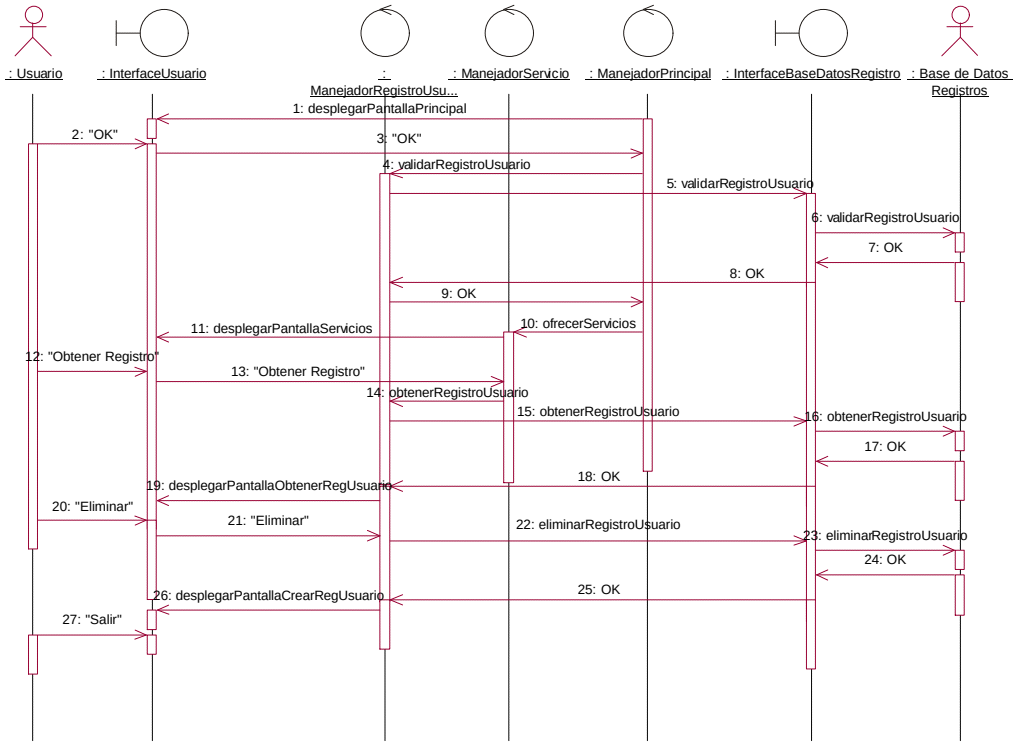

Figura 7.47 Diagrama de secuencia *Eliminar Registro Usuario* del caso de uso *Registrar Usuario*.

Registrar Tarjeta

En el caso de uso *Registrar Tarjeta* existen diversas secuencias que pueden ser instanciados por un usuario. En particular mostraremos diagramas de secuencias para *Crear Registro Tarjeta*, *Actualizar Registro Tarjeta* y *Eliminar Registro Tarjeta*. Nótese que estas secuencias incluirán obviamente a los subflujos *Crear Registro Tarjeta* (*S*-1), *Actualizar Registro Tarjeta* (*S*-4) y *Eliminar Registro Tarjeta* (*S*-5), de manera correspondiente, junto con los casos de uso *Validar Usuario* y *Ofrecer Servicios*, además de los los subflujos *Obtener Registro Usuario* (*S*-2) y *Administrar Registro Usuario* (*S*-3) de *Registrar Usuario* dado que *Registrar Tarjeta* es una extensión de este.

Crear Registro Tarjeta

El diagrama de secuencia para el subflujo *Crear Registro Tarjeta* se muestra en el diagrama 7.48. Esta secuencia inicia de manera similar a la secuencia *Actualizar Registro Usuario* con la excepción de que en lugar de hacer una actualización, se crea un registro de tarjeta siguiendo el flujo principal de *Registrar Tarjeta* en lugar del subflujo *Crear Registro Tarjeta* (*S*-1) en lugar de *Actualizar Registro Usuario* (*S*-4) en *Registrar Usuario*. Dentro del caso de uso *Registrar Tarjeta* se continuará con los subflujos *Crear Registro Tarjeta* (*S*-1), *Obtener Registro Tarjeta* (*S*-2) y *Administrar Registro Tarjeta* (*S*-3).

?? *Validar Usuario*. La secuencia comienza con la validación del usuario a partir del evento *desplegarPantallaPrincipal* de la clase *ManejadorPrincipal* a la *InterfaceUsuario*. El usuario se valida enviando el evento "*OK*". La *InterfaceUsuario* envía el mismo evento al *ManejadorPrincipal*. El *ManejadorPrincipal* envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. El *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante el mismo evento. La *InterfaceBaseDatosRegistro* envía a su vez el evento a la *Base de Datos Registros*, el cual contesta con un *OK* si la validación es buena. El *OK* es enviado a la *InterfaceBaseDatosRegistro*, de allí al *ManejadorRegistroUsuario* y luego al *ManejadorPrincipal* como respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios*.

?? *Ofrecer Servicios*. A continuación podemos pasar al caso de uso *Ofrecer Servicios* donde el *ManejadorServicio* solicita entonces a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "*Obtener Registro*". Este evento es enviado de la *InterfaceUsuario* al *ManejadorServicio* mediante el mismo evento. El evento *obtenerRegistroUsuario* es luego enviado por el *ManejadorServicio* al *ManejadorRegistroUsuario*.

?? *Registrar Usuario* subflujo *Obtener Registro Usuario* (*S*-2). A continuación regresamos al caso de uso *Registrar Usuario* subflujo *Obtener Registro Usuario* (*S*-2) donde el *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que obtenga el registro correspondiente mediante el mismo evento. Nuevamente, la *InterfaceBaseDatosRegistro* le pasa un evento similar al actor *Base de Datos Registros*, el cual contesta con un *OK*. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorRegistroUsuario*.

?? *Registrar Usuario* subflujo *Administrar Registro Usuario* (*S*-3). A continuación podemos pasar al subflujo *Administrar Registro Usuario* (*S*-3) donde el *ManejadorRegistroUsuario* solicita a la *InterfaceUsuario* desplegar la pantalla de información de registro mediante el evento *desplegarPantallaObtenerRegUsuario*. Hasta aquí la secuencia es similar al subflujo *Actualizar Registro Usuario*. En este momento el usuario oprime el botón "*Registrar Tarjeta*".

?? *Registrar Tarjeta* subflujo principal. Siguiendo con la lógica, la *InterfaceUsuario* envía el mismo evento al *ManejadorRegistroUsuario*, el cual solicita *obtenerRegistroTarjeta* al *ManejadorRegistroTarjeta*.

?? *Registrar Tarjeta* subflujo *Obtener Registro Tarjeta* (*S*-2). El *ManejadorRegistroTarjeta* que es responsable de todo lo relacionado con el registro de tarjeta, solicita *obtenerRegistroTarjeta* a la *InterfaceBaseDatosRegistro*. La *InterfaceBaseDatosRegistro* envía el mismo evento al actor *Base de Datos Registro*s. Este actor responde mediante un *NULL* correspondiente a un registro de tarjeta inexistente. Este evento es enviado de regreso al *ManejadorRegistroTarjeta*.

?? *Registrar Tarjeta* subflujo *Crear Registro Tarjeta* (*S*-1). A continuación, el *ManejadorRegistroTarjeta* solicita *desplegarPantallaCrearRegistroTarjeta* a *InterfaceUsuario*. El usuario registra sus datos de tarjeta y presiona el botón de "Registrar" generando el evento *Registrar*. Como consecuencia de esto, *InterfaceUsuario* envía el mismo evento al *ManejadorRegistroTarjeta* el cual envía el evento *crearRegistroTarjeta* a *InterfaceBaseDatosRegistro*. Este último envía el mismo evento al actor *Base de Datos Registro*s, el cual contesta con un *OK* que es sucesivamente enviado de regreso a la *InterfaceBaseDatosRegistro* para ser luego enviado al *ManejadorRegistroTarjeta*.

?? *Registrar Tarjeta* subflujo *Adminsitrar Registro Tarjeta* (*S*-3). A continuación, el *ManejadorRegistroTarjeta* envía el evento *desplegarPantallaObtenerRegTarjeta* a la *InterfaceUsuario*. En ese momento el *Usuario* presiona "*Salir*", dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios*, para luego continuar en el caso de uso *Registrar Usuario* con los subflujos *Obtener Registro Usuario* (*S*-2), *Administrar Registro Usuario* (*S*-3) y finalmente el flujo principal de *Registrar Tarjeta* seguido por los subflujos *Crear Registro Tarjeta* (*S*-1), *Obtener Registro Tarjeta* (*S*-2) y *Administrar Registro Tarjeta* (*S*-3).

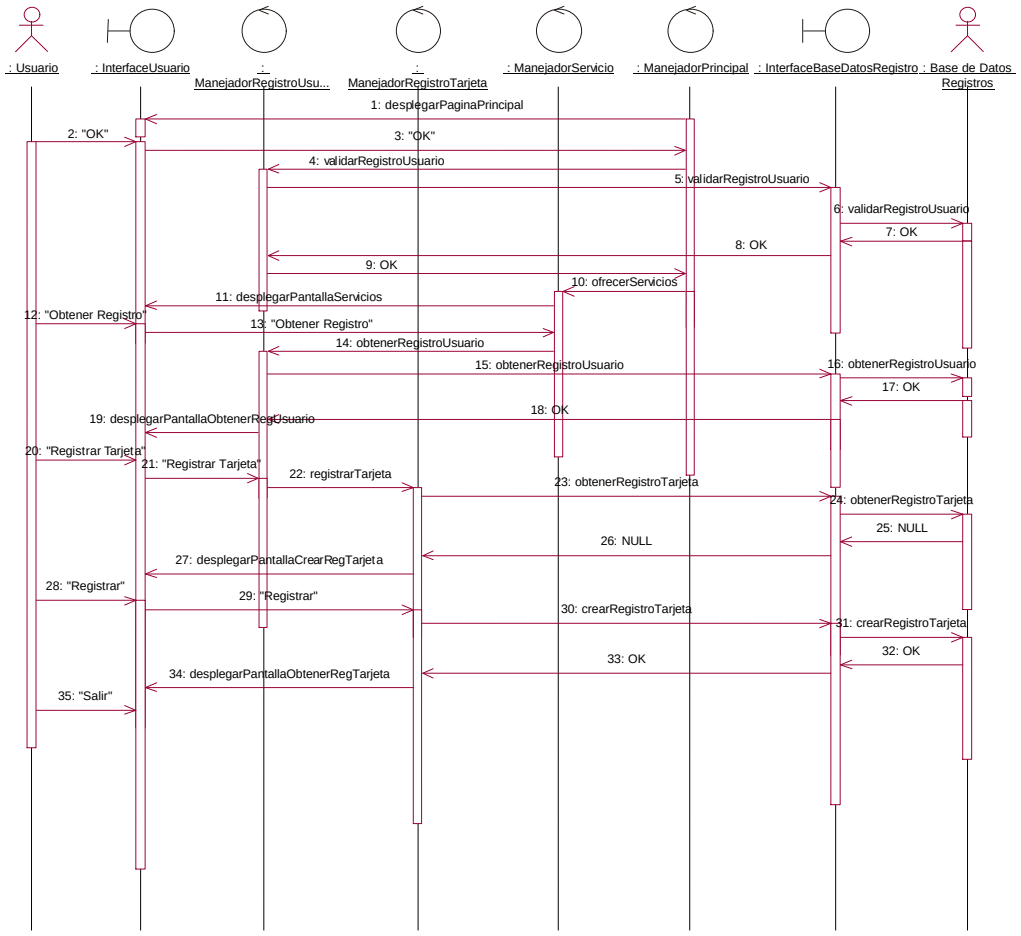

Figura 7.48 Diagrama de secuencia *Crear Registro Tarjeta* del caso de uso *Registrar Tarjeta*.

Actualizar Registro Tarjeta

El diagrama de secuencia *Actualizar Registro Tarjeta* se muestra en el diagrama 7.49. Esta secuencia es muy similar a *Crear Registro Tarjeta* aunque en lugar de seguir al subflujo *Crear Registro Tarjeta* (*S*-1), seguirá los subflujos *Obtener Registro Tarjeta* (*S*-2), *Administrar Registro Tarjeta* (*S*-3) y obviamente *Actualizar Registro Tarjeta* (*S*-4).

?? *Validar Usuario.* La secuencia comienza con la validación del usuario a partir del evento *desplegarPantallaPrincipal* de la clase *ManejadorPrincipal* a la *InterfaceUsuario*. El usuario se valida enviando el evento "*OK*". La *InterfaceUsuario* envía el mismo evento al *ManejadorPrincipal*. El *ManejadorPrincipal* envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. El *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante el mismo evento. La *InterfaceBaseDatosRegistro* envía a su vez el evento a la *Base de Datos Registros*, el cual contesta con un *OK* si la validación es buena. El *OK* es enviado a la *InterfaceBaseDatosRegistro*, de allí a *ManejadorRegistroUsuario* y luego a *ManejadorPrincipal* como respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios.*

?? *Ofrecer Servicios*. A continuación el *ManejadorServicio* solicita entonces a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "*Obtener Registro*". Este mismo evento es enviado de la *InterfaceUsuario* al *ManejadorServicio*. El evento *obtenerRegistroUsuario* es luego enviado por el *ManejadorServicio* al *ManejadorRegistroUsuario.*

?? *Registrar Usuario* subflujo *Obtener Registro Usuario* (*S*-2). A continuación regresamos al caso de uso *Registrar Usuario* subflujo *Obtener Registro Usuario* (*S*-2) donde el *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que obtenga el registro correspondiente mediante el mismo evento. Nuevamente, la *InterfaceBaseDatosRegistro* le pasa un evento similar al actor *Base de Datos Registros*, el cual contesta con un *OK*. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorRegistroUsuario*.

?? *Registrar Usuario* subflujo *Administrar Registro Usuario* (*S*-3). A continuación podemos pasar al subflujo *Administrar Registro Usuario* (*S*-3) donde el *ManejadorRegistroUsuario* solicita a la *InterfaceUsuario* desplegar la pantalla de información de registro mediante el evento *desplegarPantallaObtenerRegUsuario*. En este momento el usuario oprime el botón "*Registrar Tarjeta*".

?? *Registrar Tarjeta* subflujo principal. Siguiendo con la lógica, la *InterfaceUsuario* envía el mismo evento al *ManejadorRegistroUsuario*, el cual solicita *registrarTarjeta* al *ManejadorRegistroTarjeta*.

?? *Registrar Tarjeta* subflujo *Obtener Registro Tarjeta* (*S*-2). El *ManejadorRegistroTarjeta* solicita *obtenerRegistroTarjeta* a la *InterfaceBaseDatosRegistro*. La *InterfaceBaseDatosRegistro* envía el mismo evento al actor *Base de Datos Registro*s. El actor *Base de Datos Registros* responde mediante un *OK*, correspondiente a un registro de tarjeta existente. Este evento es enviado de regreso al *ManejadorRegistroTarjeta*.

?? *Registrar Tarjeta* subflujo *Adminsitrar Registro Tarjeta* (*S*-3). A continuación, el *ManejadorRegistroTarjeta* envía el evento *desplegarPantallaObtenerRegistroTarjeta* a *InterfaceUsuario*. El usuario actualiza sus datos de tarjeta y presiona el botón de " Actualizar" generando el evento "*Actualizar*".

?? *Registrar Tarjeta* subflujo *Actualizar Registro Tarjeta* (*S*-4). A continuación, a *InterfaceUsuario* envía el mismo evento al *ManejadorRegistroTarjeta* el cual solicita *actualizarRegistroTarjeta* a *InterfaceBaseDatosRegistro*. Este último envía el mismo evento al actor *Base de Datos Registros*, el cual contesta con un *OK* que es sucesivamente enviado de regreso a la *InterfaceBaseDatosRegistro* y luego al *ManejadorRegistroTarjeta*.

?? *Registrar Tarjeta* subflujo *Adminsitrar Registro Tarjeta* (*S*-3). A continuación, el *ManejadorRegistroTarjeta* envía el evento *desplegarPantallaObtenerRegTarjeta* a la *InterfaceUsuario*. En ese momento el *Usuario* presiona "*Salir*", dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios*, para luego continuar en el caso de uso *Registrar Usuario* con los subflujos *Obtener Registro Usuario* (*S*-2), *Administrar Registro Usuario* (*S*-3) y finalmente el flujo principal de *Registrar Tarjeta* seguido por los subflujos *Obtener Registro Tarjeta* (*S*-2), *Administrar Registro Tarjeta* (*S*-3) y *Actualizar Registro Tarjeta* (*S*-4).

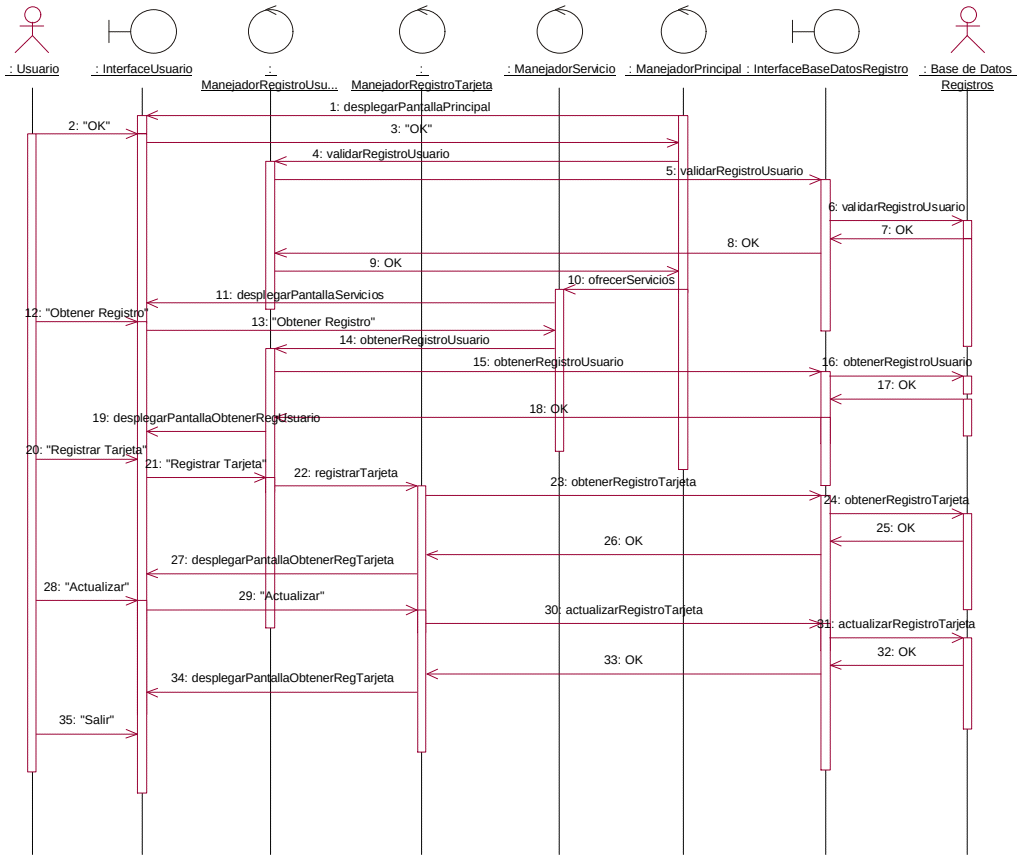

Figura 7.49 Diagrama de secuencia *Actualizar Registro Tarjeta* del caso de uso *Registrar Tarjeta*.

Eliminar Registro Tarjeta

El diagrama de secuencia *Eliminar Registro Tarjeta* se muestra en el diagrama 7.50. Esta secuencia es muy similar a *Actualizar Registro Tarjeta* aunque en lugar de seguir al subflujo *Crear Registro Tarjeta* (*S*-1), seguirá los subflujos *Obtener Registro Tarjeta* (*S*-2), *Administrar Registro Tarjeta* (*S*-3) y obviamente *Eliminar Registro Tarjeta* (*S*-5).

?? *Validar Usuario*. La secuencia comienza con la validación del usuario a partir del evento *desplegarPantallaPrincipal* de la clase *ManejadorPrincipal* al *InterfaceUsuario*. El usuario se valida enviando el evento "*OK*". La *InterfaceUsuario* envía el mismo evento al *ManejadorPrincipal*. El *ManejadorPrincipal* envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. El *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante el mismo evento. La *InterfaceBaseDatosRegistro* envía a su vez el evento a la *Base de Datos Registros*, el cual contesta con un *OK* si la validación es buena. El *OK* es enviado a la *InterfaceBaseDatosRegistro*, de allí a *ManejadorRegistroUsuario* y luego a *ManejadorPrincipal* como respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios*.

?? *Ofrecer Servicios*. A continuación el *ManejadorServicio* solicita entonces a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "*Obtener Registro*". Este mismo evento es enviado de la *InterfaceUsuario* al *ManejadorServicio*. El evento *obtenerRegistroUsuario* es luego enviado por el *ManejadorServicio* al *ManejadorRegistroUsuario*.

?? *Registrar Usuario* subflujo *Obtener Registro Usuario* (*S*-2). A continuación regresamos al caso de uso *Registrar Usuario* subflujo *Obtener Registro Usuario* (*S*-2) donde el *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que obtenga el registro correspondiente mediante el mismo evento. Nuevamente, la *InterfaceBaseDatosRegistro* le pasa un evento similar al actor *Base de Datos Registros*, el cual contesta con un *OK*. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorRegistroUsuario*.

?? *Registrar Usuario* subflujo *Administrar Registro Usuario* (*S*-3). A continuación podemos pasar al subflujo *Administrar Registro Usuario* (*S*-3) donde el *ManejadorRegistroUsuario* solicita a la *InterfaceUsuario* desplegar la pantalla de información de registro mediante el evento *desplegarPantallaObtenerRegUsuario*. En este momento el usuario oprime el botón "*Registrar Tarjeta*".

?? *Registrar Tarjeta* subflujo principal. Siguiendo con la lógica, la *InterfaceUsuario* envía el mismo evento al *ManejadorRegistroUsuario*, el cual envía el evento *registrarTarjeta* al *ManejadorRegistroTarjeta*.

?? *Registrar Tarjeta* subflujo *Obtener Registro Tarjeta* (*S*-2). El *ManejadorRegistroTarjeta* solicita *obtenerRegistroTarjeta* a la *InterfaceBaseDatosRegistro*. La *InterfaceBaseDatosRegistro* envía el mismo evento al actor *Base de Datos Registro*s. El actor *Base de Datos Registro*s responde mediante un *OK* correspondiente a un registro de tarjeta existente. Este evento es enviado de regreso al *ManejadorRegistroTarjeta*.

?? *Registrar Tarjeta* subflujo *Adminsitrar Registro Tarjeta* (*S*-3). A continuación, el *ManejadorRegistroTarjeta* envía el evento *desplegarPantallaObtenerRegistroTarjeta* a *InterfaceUsuario*. Hasta aquí la secuencia es similar al subflujo *Actualizar Registro Tarjeta*, ya que el usuario presiona el botón de "Eliminar" generando el evento "*Eliminar*".

?? *Registrar Tarjeta* subflujo *Eliminar Registro Tarjeta* (*S*-5). Como consecuencia del evento "Eliminar", la *InterfaceUsuario* mismo evento al *ManejadorRegistroTarjeta* el cual solicita *registrarTarjeta* al *ManejadorRegistroTarjeta* el cual envía el evento *eliminarRegistroTarjeta* a *InterfaceBaseDatosRegistro*. Este último envía el mismo evento al actor *Base de Datos Registro*s, el cual contesta con un *OK* que es sucesivamente enviado de regreso a la *InterfaceBaseDatosRegistro* y luego al *ManejadorRegistroTarjeta*.

?? *Registrar Tarjeta* subflujo *Adminsitrar Registro Tarjeta* (*S*-3). A continuación, el *ManejadorRegistroTarjeta* envía el evento *desplegarPantallaCrearRegTarjeta* a la *InterfaceUsuario*. En ese momento el *Usuario* presiona *Salir*, dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios*, para luego continuar en el caso de uso *Registrar Usuario* con los subflujos *Obtener Registro Usuario* (*S*-2), *Administrar Registro Usuario* (*S*-3) y finalmente el flujo principal de *Registrar Tarjeta* seguido por los subflujos *Obtener Registro Tarjeta* (*S*-2), *Administrar Registro Tarjeta* (*S*-3) y *Eliminar Registro Tarjeta* (*S*-5).

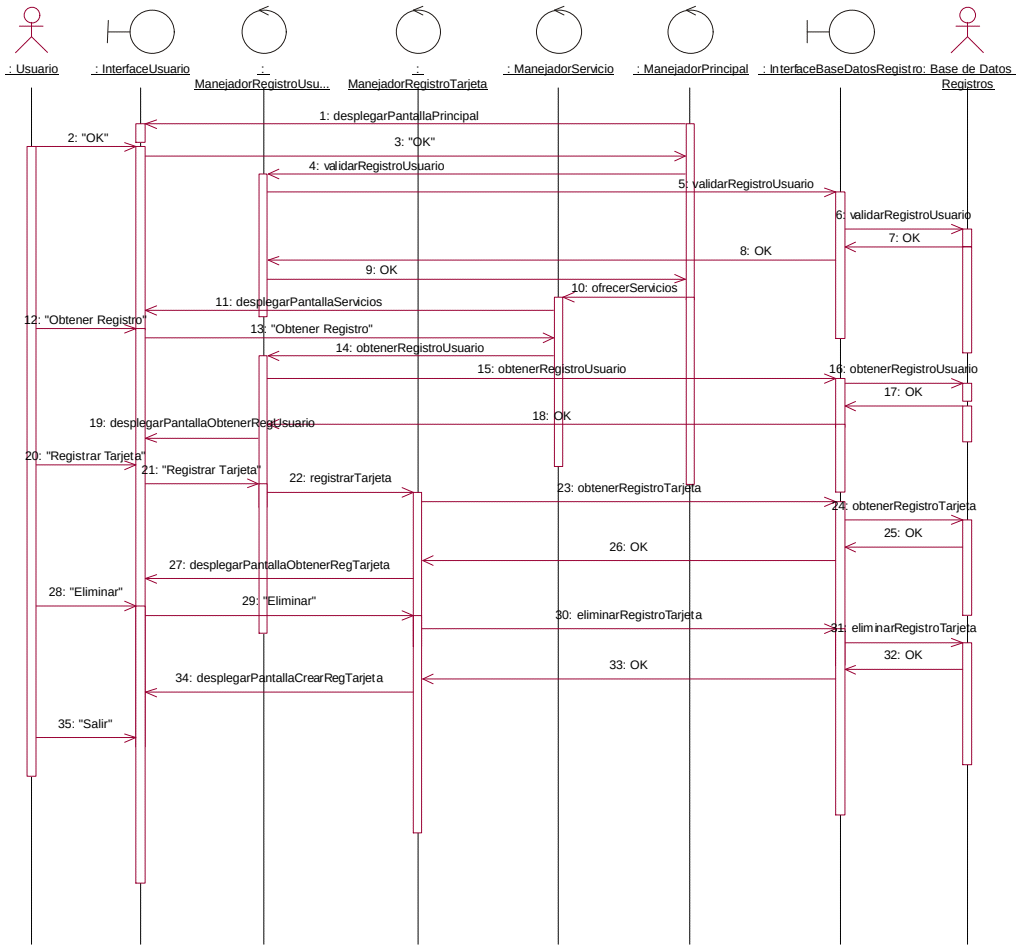

Figura 7.50 Diagrama de secuencia *Eliminar Registro Tarjeta* del caso de uso *Registrar Tarjeta*.

Consultar Información

En el caso de uso *Consultar Información* existen diversos subflujos que pueden ser instanciados por un usuario. En particular mostraremos diagramas de secuencias para *Consultar Horarios*, *Consultar Tarifas* y *Consultar Estado*. Nótese que estas secuencias incluirán obviamente a los subflujos *Consultar Horarios* (*S*-2), *Consultar Tarifas* (*S*-3) y *Consultar Estado* (*S*-4), respectivamente. Además se incluirá parte de los casos de uso de inclusión *Validar Usuario* y *Ofrecer Servicios*, además de subflujos adicionales dentro del caso de uso *Consultar Información*.

Consultar Horarios

El diagrama de secuencia *Consultar Horarios* se muestra en el diagrama 7.51. Esta secuencia inicia en el flujo principal de *Consultar Información* que incluye la validación del usuario en *Validar Usuario* seguidos por *Ofrecer Servicios* y los subflujos *Consultar* (*S*-1), *Consultar Horarios* (*S*-2) y *Devolver Horarios* (*S*-3).

?? *Validar Usuario.* Esta secuencia comienza con la validación del usuario a partir del evento *desplegarPantallaPrincipal* de la clase *ManejadorPrincipal* a la *InterfaceUsuario*. El usuario se valida enviando el evento "*OK*". La *InterfaceUsuario* envía el mismo evento al *ManejadorPrincipal*. El *ManejadorPrincipal* envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. El *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante el mismo evento. La *InterfaceBaseDatosRegistro* envía a su vez el evento a la *Base de Datos Registros*, el cual contesta con un *OK* si la validación es buena. El *OK* es enviado a la *InterfaceBaseDatosRegistro*, de allí a *ManejadorRegistroUsuario* y luego a *ManejadorPrincipal* como

respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios*.

?? *Ofrecer Servicios*. A continuación podemos pasar al caso de uso *Ofrecer Servicios* donde el *ManejadorServicio* solicita entonces a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "Consultar Información". Se envía el mismo evento de la *InterfaceUsuario* al *ManejadorServicio*. El evento *consultarInformación* es luego enviado por el *ManejadorServicio* al *ManejadorConsultas*.

?? *Consultar Información* subflujo *Consultar* (*S*-1). A continuación el *ManejadorConsultas* envía el evento *desplegarPantallaConsultas* a *InterfaceUsuario*. El usuario presiona "Horarios" lo cual genera que la *InterfaceUsuario* envíe este evento de regreso a *ManejadorConsultas* el cual envía el evento *consultarHorarios* al *ManejadorConsultaHorarios*.

?? *Consultar Información* subflujo *Consultar Horarios* (*S*-2). A continuación el *ManejadorConsultaHorarios* envía el evento *desplegarPantallaConsultaHorarios* a *InterfaceUsuario*. El usuario llena la información de la consulta y oprime el botón "Consultar" el cual genera el mismo evento de *InterfaceUsuario* al *ManejadorConsultaHorarios*. Este último envía el evento *consultarHorarios* a la *InterfaceBaseDatosReservas* para que haga la petición correspondiente al actor *Base de Datos Reservas*, el cual contesta con un *OK*. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorConsultaHorarios*.

?? *Consultar Información* subflujo *Devolver Horarios* (*S*-3). A continuación el *ManejadorConsultaHorarios* envía el evento *desplegarPantallaResultadoHorarios* a la *InterfaceUsuario* para desplegar los resultados de la búsqueda. En ese momento el *Usuario* presiona "*Salir*", dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios*, para luego continuar en el caso de uso *Consultar Información* con los subflujos *Consultar* (*S*-1), *Consultar Horarios* (*S*-2) y *Devolver Horarios* (*S*-3).

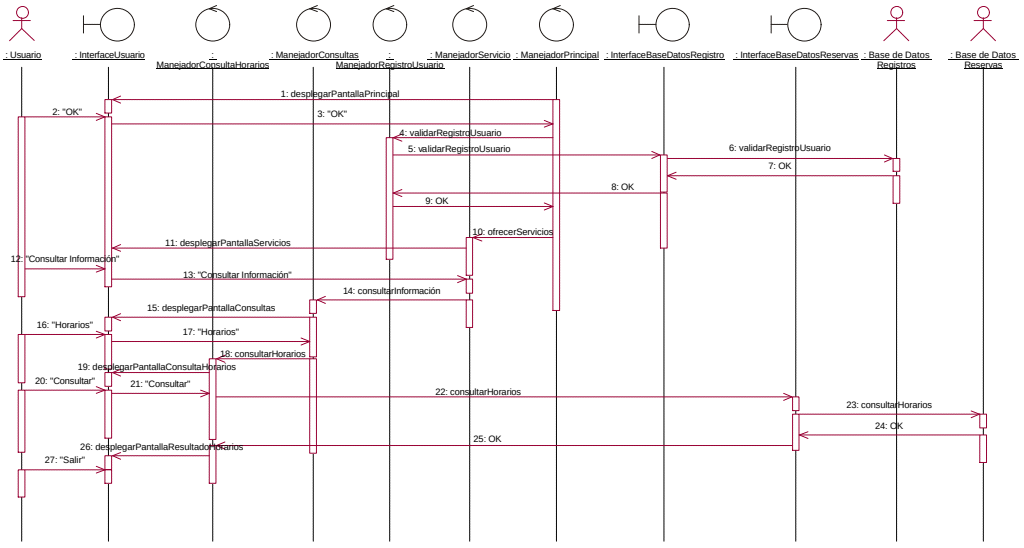

Figura 7.51 Diagrama de secuencia *Consultar Horarios* del caso de uso *Consultar Información*.

Consultar Tarifas

El diagrama de secuencia *Consultar Tarifas* se muestra en el diagrama 7.52. Esta secuencia inicia en el flujo principal de *Consultar Información* que incluye la validación del usuario en *Validar Usuario* seguidos por *Ofrecer Servicios* y los subflujos *Consultar* (*S*-1), *Consultar Tarifas* (*S*-4) y *Devolver Tarifas* (*S*-5).

?? *Validar Usuario*. Esta secuencia comienza con la validación del usuario a partir del evento *desplegarPantallaPrincipal* de la clase *ManejadorPrincipal* a la *InterfaceUsuario*. El usuario se valida enviando el evento "*OK*". La *InterfaceUsuario* envía el mismo evento al *ManejadorPrincipal*. El *ManejadorPrincipal* envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. El *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante el mismo evento. La *InterfaceBaseDatosRegistro* envía a su vez el evento a la *Base de Datos*

Registros, el cual contesta con un *OK* si la validación es buena. El *OK* es enviado a la *InterfaceBaseDatosRegistro*, de allí a *ManejadorRegistroUsuario* y luego a *ManejadorPrincipal* como respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios*.

?? *Ofrecer Servicios*. A continuación podemos pasar al caso de uso *Ofrecer Servicios* donde el *ManejadorServicio* solicita a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "Consultar Información". Se envía el mismo evento de la *InterfaceUsuario* al *ManejadorServicio*. El evento *consultarInformación* es luego enviado por el *ManejadorServicio* al *ManejadorConsultas.*

?? *Consultar Información* subflujo *Consultar* (*S*-1). A continuación el *ManejadorConsultas* envía el evento *desplegarPantallaConsultas* a la *InterfaceUsuario*. Hasta aquí la secuencia es similar a la secuencia *Consultar Horarios*. En esta nueva secuencia, el usuario presiona "Tarifas" lo cual genera que la *InterfaceUsuario* envíe este evento de regreso a *ManejadorConsultas* el cual envía el evento *consultarTarifas* al *ManejadorConsultaTarifas*.

?? *Consultar Información* subflujo *Consultar Tarifas* (*S*-4). A continuación el *ManejadorConsultaTarifas* envía el evento *desplegarPantallaConsultaTarifas* a *InterfaceUsuario*. El usuario llena la información de la consulta y oprime el botón "Consultar" el cual genera el mismo evento de *InterfaceUsuario* al *ManejadorConsultaTarifas*. Este último envía el evento *consultarTarifas* a la *InterfaceBaseDatosReservas* para que haga la petición correspondiente al actor *Base de Datos Reservas*, el cual contesta con un *OK*. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorConsultaTarifas.*

?? *Consultar Información* subflujo *Devolver Tarifas* (*S*-5). A continuación el *ManejadorConsultaTarifas* envía el evento *desplegarPantallaResultadoTarifas* a la *InterfaceUsuario* para desplegar los resultados de la búsqueda. En ese momento el *Usuario* presiona "*Salir*", dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios*, para luego continuar en el caso de uso *Consultar Información* con los subflujos *Consultar* (*S*-1), *Consultar Tarifas* (*S*-4) y *Devolver Tarifas* (*S*-5).

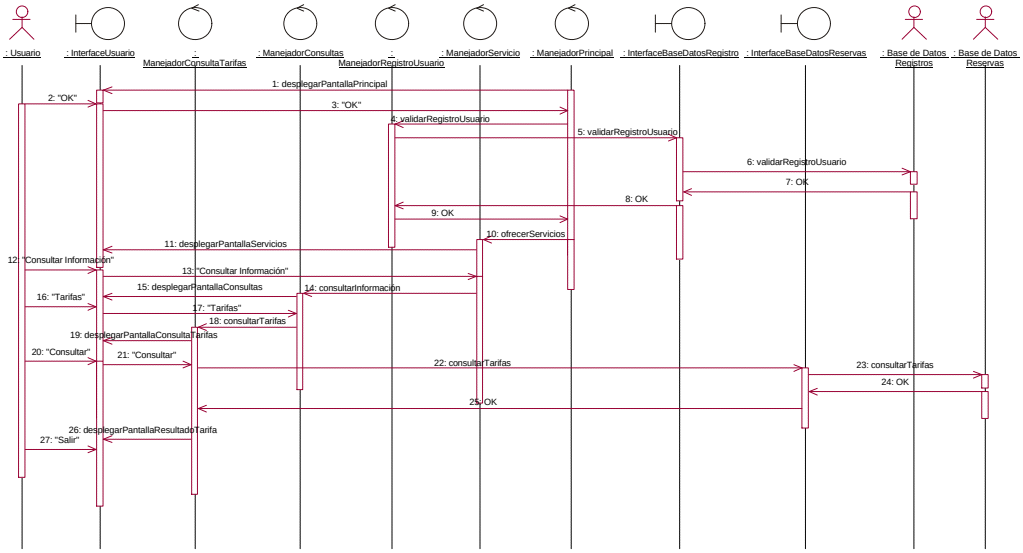

Figura 7.52 Diagrama de secuencia *Consultar Tarifas* del caso de uso *Consultar Información*.

Consultar Estado

El diagrama de secuencia *Consultar Estado* se muestra en el diagrama 7.53. Esta secuencia inicia en el flujo principal de *Consultar Información* que incluye la validación del usuario en *Validar Usuario* seguidos por *Ofrecer Servicios* y los subflujos *Consultar* (*S*-1), *Consultar Estado* (*S*-6) y *Devolver Estado* (*S*-7).

?? *Validar Usuario.* Esta secuencia comienza con la validación del usuario a partir del evento *desplegarPantallaPrincipal* de la clase *ManejadorPrincipal* a la *InterfaceUsuario*. El usuario se valida enviando el evento "*OK*". La *InterfaceUsuario* envía el mismo evento al *ManejadorPrincipal*. El

ManejadorPrincipal envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. El *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante el mismo evento. La *InterfaceBaseDatosRegistro* envía a su vez el evento a la *Base de Datos Registros*, el cual contesta con un *OK* si la validación es buena. El *OK* es enviado a la *InterfaceBaseDatosRegistro*, de allí a *ManejadorRegistroUsuario* y luego a *ManejadorPrincipal* como respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios.*

?? *Ofrecer Servicios*. A continuación podemos pasar al caso de uso *Ofrecer Servicios* donde el *ManejadorServicio* solicita a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "Consultar Información". Se envía el mismo evento de la *InterfaceUsuario* al *ManejadorServicio*. El evento *consultarInformación* es luego enviado por el *ManejadorServicio* al *ManejadorConsultas.*

?? *Consultar Información* subflujo *Consultar* (*S*-1). A continuación el *ManejadorConsultas* envía el evento *desplegarPantallaConsultas* a la *InterfaceUsuario*. Hasta aquí la secuencia es similar a la secuencia *Consultar Horarios* y *Consultar Tarifas*. En esta nueva secuencia, el usuario presiona "Estado" lo cual genera que la *InterfaceUsuario* envíe este evento de regreso a *ManejadorConsultas* el cual envía el evento *consultarEstado* al *ManejadorConsultaEstado*.

?? *Consultar Información* subflujo *Consultar Estado* (*S*-6). A continuación el *ManejadorConsultaEstado* envía el evento *desplegarPantallaConsultaEstado* a *InterfaceUsuario*. El usuario llena la información de la consulta y oprime el botón "Consultar" el cual genera el mismo evento de *InterfaceUsuario* al *ManejadorConsultaEstado*. Este último envía el evento *consultarEstados* a la *InterfaceBaseDatosReservas* para que haga la petición correspondiente al actor *Base de Datos Reservas*, el cual contesta con un *OK*. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorConsultaEstado.*

?? *Consultar Información* subflujo *Devolver Estado* (*S*-7). A continuación el *ManejadorConsultaEstado* envía el evento *desplegarPantallaResultadoEstado* a la *InterfaceUsuario* para desplegar los resultados de la búsqueda. En ese momento el *Usuario* presiona "*Salir*", dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios*, para luego continuar en el caso de uso *Consultar Información* con los subflujos *Consultar* (*S*-1), *Consultar Estado* (*S*-4) y *Devolver Estado* (*S*-7).

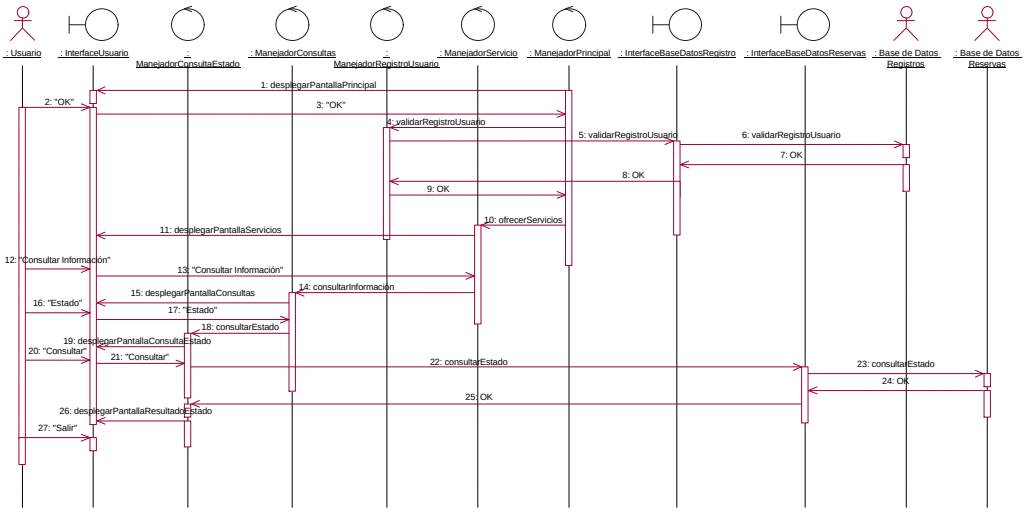

Figura 7.53 Diagrama de secuencia *Consultar Estado* del caso de uso *Consultar Información*.

Hacer Reservación

En el caso de uso *Hacer Reservación* existen diversos subflujos que pueden ser instanciados por un usuario. En particular mostraremos diagramas de secuencias para *Crear Reservación*, *Actualizar Reservación* y *Eliminar Reservación*. Nótese que estas secuencias incluirán obviamente a los subflujos *Crear Reservación* (*S*-2), *Actualizar Reservación* (*S*-5) y *Eliminar Reservación* (*S*-6), respectivamente. Además se incluirá parte de los casos de uso de

inclusión *Validar Usuario* y *Ofrecer Servicios*, además de subflujos adicionales dentro del caso de uso *Hacer Reservación*.

Crear Reservación

El diagrama de secuencia *Crear Reservación* se muestra en el diagrama 7.54. Esta secuencia inicia en el flujo principal de *Hacer Reservación* que incluye la validación del usuario en *Validar Usuario* seguidos por *Ofrecer Servicios* y los subflujos *Solicitar Clave Reservación* (*S*-1), *Crear Reservación* (*S*-2) y *Administrar Reservación* (*S*-4).

?? *Validar Usuario.* Esta secuencia comienza con la validación del usuario a partir del evento *desplegarPantallaPrincipal* de la clase *ManejadorPrincipal* a la *InterfaceUsuario*. El usuario se valida enviando el evento "*OK*". La *InterfaceUsuario* envía el mismo evento al *ManejadorPrincipal*. El *ManejadorPrincipal* envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. El *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante el mismo evento. La *InterfaceBaseDatosRegistro* envía a su vez el evento a la *Base de Datos Registros*, el cual contesta con un *OK* si la validación es buena. El *OK* es enviado a la *InterfaceBaseDatosRegistro*, de allí a *ManejadorRegistroUsuario* y luego a *ManejadorPrincipal* como respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios.*

?? *Ofrecer Servicios*. A continuación podemos pasar al caso de uso *Ofrecer Servicios* donde el *ManejadorServicio* solicita entonces a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "Hacer Reservación". Se envía el mismo evento de la *InterfaceUsuario* al *ManejadorServicio*. El *ManejadorServicio* envía el evento *reservar* al *ManejadorReservas.*

?? *Hacer Reservación* subflujo *Solicitar Clave Reservación* (*S*-1). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaClaveReservas* a *InterfaceUsuario*. El usuario presiona el botón "Crear". La *InterfaceUsuario* envía el mismo evento a *ManejadorReservas.*

?? *Hacer Reservación* subflujo *Crear Reservación* (*S*-2). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaCrearReservaVuelos* a *InterfaceUsuario*. El usuario presiona "Reservar". La *InterfaceUsuario* envía el mismo evento a *ManejadorReservas* el cual envía el evento *crearReserva* a la *InterfaceBaseDatosReservas* para que haga la petición correspondiente al actor *Base de Datos Reservas*, el cual contesta con un *OK*. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorReservas.*

?? *Hacer Reservación* subflujo *Administrar Reservación* (*S*-4). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaRecordReservaVuelos* a la *InterfaceUsuario* para desplegar los resultados de la creación. En ese momento el *Usuario* presiona "*Salir*", dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios*, para luego continuar en el caso de uso *Hacer Reservación* con los subflujos *Solicitar Clave Reservación* (*S*-1) *Crear Reservación* (*S*-2) y *Administrar Reservación* (*S*-4).

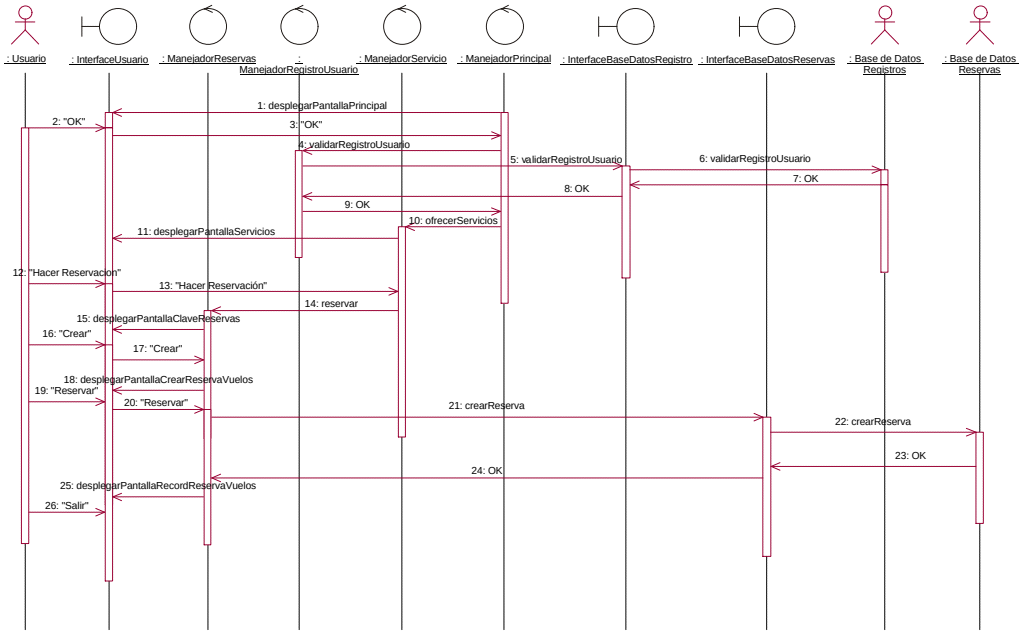

Figura 7.54 Diagrama de secuencia *Crear Reservación* del caso de uso *Hacer Reservación*.

Actualizar Reservación

El diagrama de secuencia *Actualizar Reservación* se muestra en el diagrama 7.55. Esta secuencia inicia en el flujo principal de *Hacer Reservación* que incluye la validación del usuario en *Validar Usuario* seguidos por *Ofrecer Servicios* y los subflujos *Solicitar Clave Reservación* (*S*-1), *Obtener Reservación* (*S*-3), *Administrar Reservación* (*S*-4) y *Actualizar Reservación* (*S*-5).

?? *Validar Usuario*. Esta secuencia comienza con la validación del usuario a partir del evento *desplegarPantallaPrincipal* de la clase *ManejadorPrincipal* a la *InterfaceUsuario*. El usuario se valida enviando el evento "*OK*". La *InterfaceUsuario* envía el mismo evento al *ManejadorPrincipal*. El *ManejadorPrincipal* envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. El *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante el mismo evento. La *InterfaceBaseDatosRegistro* envía a su vez el evento a la *Base de Datos Registros*, el cual contesta con un *OK* si la validación es buena. El *OK* es enviado a la *InterfaceBaseDatosRegistro*, de allí a *ManejadorRegistroUsuario* y luego a *ManejadorPrincipal* como respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios*.

?? *Ofrecer Servicios*. A continuación podemos pasar al caso de uso *Ofrecer Servicios* donde el *ManejadorServicio* solicita entonces a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "Hacer Reservación". Se genera el mismo evento que es enviado de la *InterfaceUsuario* al *ManejadorServicio*. El *ManejadorServicio* envía el evento *reservar* al *ManejadorReservas*.

?? *Hacer Reservación* subflujo *Solicitar Clave Reservación* (*S*-1). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaClaveReservas* a *InterfaceUsuario*. El usuario presiona el botón "Obtener", junto con la inserción de la clave de récord de reservas. La *InterfaceUsuario* envía el mismo evento de regreso a *ManejadorReservas*.

?? *Hacer Reservación* subflujo *Obtener Reservación* (*S*-3). A continuación el *ManejadorReservas* envía el evento *obtenerReserva* a la *InterfaceBaseDatosReservas*, la cual a su vez envía este evento al actor *Base de Datos Reservas*. El actor genera un *OK* si todo es correcto y se lo envía de regreso a *InterfaceBaseDatosReservas*, la cual luego lo envía a *ManejadorReservas*.

?? *Hacer Reservación* subflujo *Administrar Reservación* (*S*-4). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaRecordReservaVuelos* a *InterfaceUsuario*. El usuario actualiza los datos de su reservación y presiona "Actualizar".

?? *Hacer Reservación* subflujo *Actualizar Reservación* (*S*-5). A continuación la *InterfaceUsuario* envía el evento "Actualizar" de regreso a *ManejadorReservas.* El *ManejadorReservas* solicita *actualizarReserva* a la *InterfaceBaseDatosReservas* para que haga la petición correspondiente al actor *Base de Datos Reservas*, el cual contesta con un *OK*. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorReservas.*

?? *Hacer Reservación* subflujo *Administrar Reservación* (*S*-4). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaRecordReservaVuelos* a la *InterfaceUsuario* para desplegar los resultados de la actualización. En ese momento el *Usuario* presiona "*Salir*", dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios*, para luego continuar en el caso de uso *Hacer Reservación* con los subflujos *Solicitar Clave Reservación* (*S*-1), *Obtener Reservación* (*S*-3), *Administrar Reservación* (*S*-4) y *Actualizar Reservación* (*S*-5).

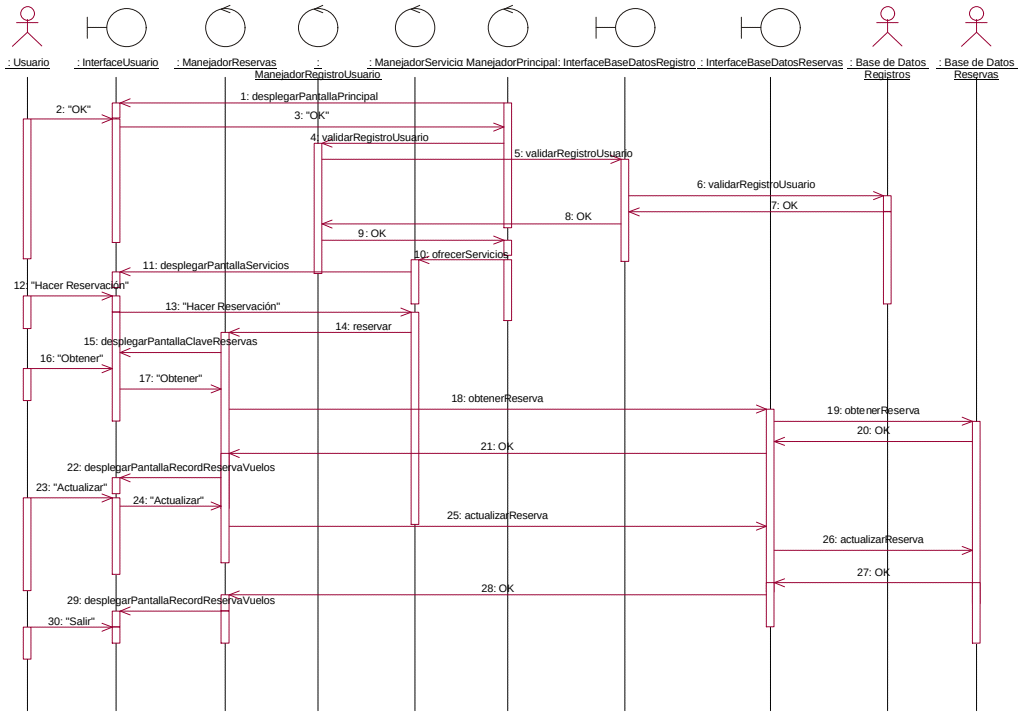

Figura 7.55 Diagrama de secuencia *Actualizar Reservación* del caso de uso *Hacer Reservación*.

Eliminar Reservación

El diagrama de secuencia *Eliminar Reservación* se muestra en el diagrama 7.56. Esta secuencia inicia en el flujo principal de *Hacer Reservación* que incluye la validación del usuario en *Validar Usuario* seguidos por *Ofrecer Servicios* y los subflujos *Solicitar Clave Reservación* (*S*-1), *Obtener Reservación* (*S*-3), *Administrar Reservación* (*S*-4) y *Eliminar Reservación* (*S*-6).

?? *Validar Usuario.* Esta secuencia comienza con la validación del usuario a partir del evento *desplegarPantallaPrincipal* de la clase *ManejadorPrincipal* a la *InterfaceUsuario*. El usuario se valida enviando el evento "*OK*". La *InterfaceUsuario* envía el mismo evento al *ManejadorPrincipal*. El *ManejadorPrincipal* envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. El *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante el mismo evento. La *InterfaceBaseDatosRegistro* envía a su vez el evento a la *Base de Datos Registros*, el cual contesta con un *OK* si la validación es buena. El *OK* es enviado a la *InterfaceBaseDatosRegistro*, de allí a *ManejadorRegistroUsuario* y luego a *ManejadorPrincipal* como

respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios.*

?? *Ofrecer Servicios*. A continuación podemos pasar al caso de uso *Ofrecer Servicios* donde el *ManejadorServicio* solicita a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "Hacer Reservación". Se envía el mismo evento de la *InterfaceUsuario* al *ManejadorServicio*. El *ManejadorServicio* envía el evento *reservar* al *ManejadorReservas*.

?? *Hacer Reservación* subflujo *Solicitar Clave Reservación* (*S*-1). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaClaveReservas* a *InterfaceUsuario*. El usuario presiona el botón "Obtener", junto con la inserción de la clave de récord de reservas. La *InterfaceUsuario* envía el mismo evento de regreso a *ManejadorReservas.*

?? *Hacer Reservación* subflujo *Obtener Reservación* (*S*-3). A continuación el *ManejadorReservas* envía el evento *obtenerReserva* a la *InterfaceBaseDatosReservas,* la cual a su vez envía este evento al actor *Base de Datos Reservas*. El actor genera un *OK* si todo es correcto y se lo envía de regreso a *InterfaceBaseDatosReservas*, la cual luego lo envía a *ManejadorReservas.*

?? *Hacer Reservación* subflujo *Administrar Reservación* (*S*-4). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaRecordReservaVuelos* a *InterfaceUsuario*. Hasta aquí la secuencia es similar a *Actualizar Reservación*. En este momento el usuario presiona "Eliminar".

?? *Hacer Reservación* subflujo *Eliminar Reservación* (*S*-6). A continuación la *InterfaceUsuario* envía el mismo evento de regreso a *ManejadorReservas* el cual envía el evento *eliminarReserva* a la *InterfaceBaseDatosReservas* para que haga la petición correspondiente al actor *Base de Datos Reservas*, el cual contesta con un *OK*. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorReservas*

?? *Hacer Reservación* subflujo *Crear Reservación* (*S*-2). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaCrearReservaVuelos* a la *InterfaceUsuario* para desplegar una nueva pantalla de creación de reservas. En ese momento el *Usuario* presiona "*Salir*", dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios,* para luego continuar en el caso de uso *Hacer Reservación* con los subflujos *Solicitar Clave Reservación* (*S*-1), *Crear Reservación* (*S*-2), *Obtener Reservación* (*S*-3), *Administrar Reservación* (*S*-4) y *Eliminar Reservación* (*S*-6).

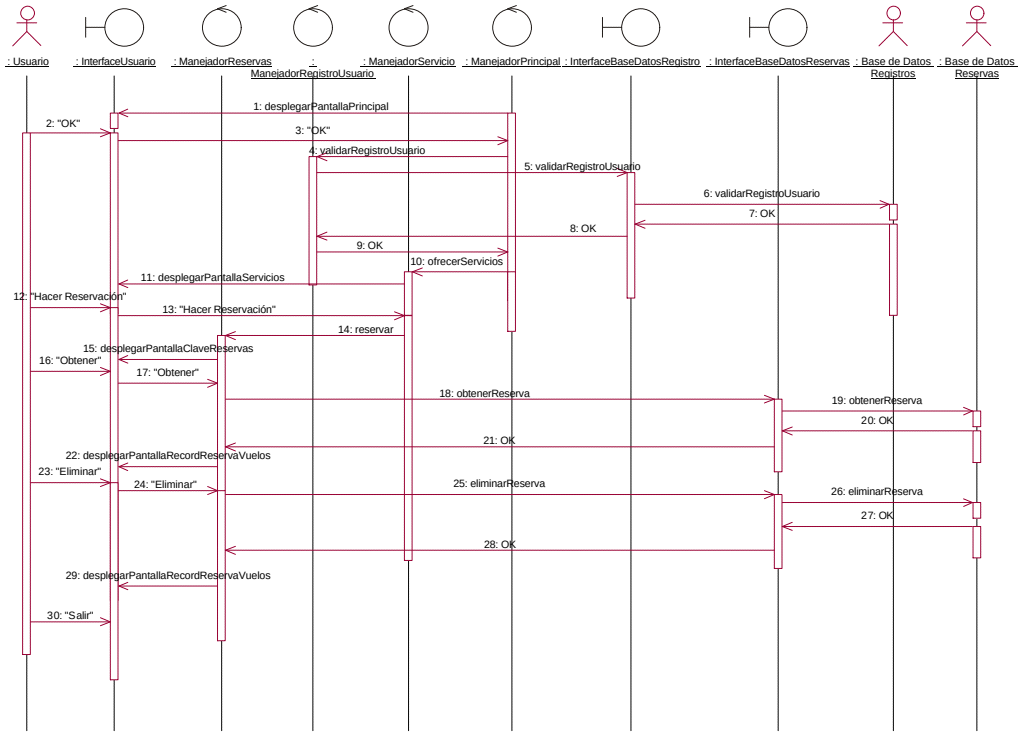

Figura 7.56 Diagrama de secuencia *Eliminar Reservación* del caso de uso *Hacer Reservación*.

Pagar Reservación

En el caso de uso *Pagar Reservación* existen diversos subflujos que pueden ser instanciados por un usuario. En particular mostraremos diagramas de secuencias para *Pagar Reservación* y *Reembolsar Pago*. Nótese que estas secuencias incluirán obviamente a los subflujos *Pagar Reservación* (*S*-1) y *Reembolsar Pago* (*S*-2), respectivamente. Además se incluirá parte de los casos de uso de inclusión *Validar Usuario* y *Ofrecer Servicios*, los subflujos, *Solicitar Clave Reservación* (*S*-1), *Obtener Reservación* (*S*-3) y *Administrar Reservación* (*S*-4), necesarios por extensión de *Hacer Reservación*, además de subflujos adicionales dentro del caso de uso *Pagar Reservación*.

Pagar Reservación

El diagrama de secuencia *Pagar Reservación* se muestra en el diagrama 7.57. Esta secuencia inicia en el flujo principal de *Pagar Reservación* que incluye, por ser una extensión, la validación del usuario en *Validar Usuario* seguidos por *Ofrecer Servicios*, los subflujos *Solicitar Clave Reservación* (*S*-1), *Obtener Reservación* (*S*-3) y *Administrar Reservación* (*S*-4), del caso de uso *Hacer Reservación*. Posteriormente se ejecuta el flujo principal de *Pagar Reservación* junto con el subflujo *Pagar Reservación* (*S*-1). Además de esto, se ejecuta el subflujo *Obtener Registro Tarjeta* (*S*-2) del caso de uso *Registrar Tarjeta*.

?? *Validar Usuario*. Esta secuencia comienza comienza con la validación del usuario a partir del evento *desplegarPantallaPrincipal* de la clase *ManejadorPrincipal* a la *InterfaceUsuario*. El usuario se valida enviando el evento "*OK*". La *InterfaceUsuario* envía el mismo evento al *ManejadorPrincipal*. El *ManejadorPrincipal* envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. El *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante el mismo evento. La *InterfaceBaseDatosRegistro* envía a su vez el evento a la *Base de Datos Registros*, el cual contesta con un *OK* si la validación es buena. El *OK* es enviado a la *InterfaceBaseDatosRegistro*, de allí a *ManejadorRegistroUsuario* y luego a *ManejadorPrincipal* como respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios*.

?? *Ofrecer Servicios*. A continuación podemos pasar al caso de uso *Ofrecer Servicios* donde el *ManejadorServicio* solicita entonces a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "Hacer Reservación" y debe incluir un número correspondiente a la clave de reservación. Se envía el mismo evento de la *InterfaceUsuario* al *ManejadorServicio*. El *ManejadorServicio* envía el evento *reservar* al *ManejadorReservas*.

?? *Hacer Reservación* subflujo *Solicitar Clave Reservación* (*S*-1). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaClaveReservas* a *InterfaceUsuario*. El usuario presiona el botón "Obtener", junto con la inserción de la clave de récord de reservas. La *InterfaceUsuario* envía el mismo evento de regreso a *ManejadorReservas*.

?? *Hacer Reservación* subflujo *Obtener Reservación* (*S*-3). A continuación el *ManejadorReservas* envía el evento *obtenerReserva* a la *InterfaceBaseDatosReservas*, la cual a su vez envía este evento al actor *Base de Datos Reservas*. El actor genera un *OK* si todo es correcto y se lo envía de regreso a *InterfaceBaseDatosReservas*, la cual luego lo envía a *ManejadorReservas*.

?? *Hacer Reservación* subflujo *Administrar Reservación* (*S*-4). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaRecordReservaVuelos* a *InterfaceUsuario*. Hasta aquí la secuencia es similar a *Actualizar Reservación* y *Eliminar Reservación* del caso de uso *Hacer Reservación*. En este momento el usuario presiona "Pagar". La *InterfaceUsuario* envía el mismo evento de regreso a *ManejadorReservas* el cual envía el evento *pagarReserva* al *ManejadorPagos*.

?? *Pagar Reservación* flujo principal. A continuación el *ManejadorPagos* envía el evento *obtenerRegistroTarjeta* al *ManejadorRegistroTarjeta*.

?? *Registrar Tarjeta* subflujo *Obtener Registro Tarjeta* (*S*-2). A continuación el *ManejadorRegistroTarjeta* envía el evento *obtenerRegistroTarjeta* a la *InterfaceBaseDatosRegistro* la cual envía el mismo evento al actor *Base de Datos Registros*. Este actor regresa un *OK* a la *InterfaceBaseDatosRegistro* la cual a su vez lo envía de regreso a *ManejadorRegistroTarjeta*.

?? *Pagar Reservación* flujo principal (continuación anterior). A continuación el *ManejadorRegistroTarjeta* envía el *OK* a *ManejadorPagos*.

?? *Pagar Reservación* subflujo *Pagar Reservación* (*S*-1). A continuación el *ManejadorPagos* envía el evento *desplegarPantallaPagarRegTarjeta* a la *InterfaceUsuario*. El usuario genera el evento "*Pagar*". La *InterfaceUsuario* recibe la petición y envía el mismo evento al *ManejadorPagos*. Este a su vez envía el evento

pagarReserva a la *InterfaceBaseDatosReservas* para que haga la petición correspondiente al actor *Base de Datos Reservas*, el cual contesta con un *OK*. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorPagos* el cual envía el *OK* al *ManejadorReservas*.

?? *Hacer Reservación* subflujo *Administrar Reservación* (*S*-4). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaRecordReservaVuelos* a la *InterfaceUsuario* para desplegar los resultados del pago de reserva. En ese momento el *Usuario* presiona "*Salir*", dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios*, para luego continuar en el caso de uso *Hacer Reservación* con los subflujos *Solicitar Clave Reservación* (*S*-1), *Obtener Reservación* (*S*-3) y *Administrar Reservación* (*S*-4), y finalizar en el caso de uso *Pagar Reservación* con el flujo principal y el subflujo *Pagar Reservación* (*S*-1). Se incluye también el caso de uso *Registrar TArjeta* subflujo *Obtener Registro Tarjeta* (*S*-2).

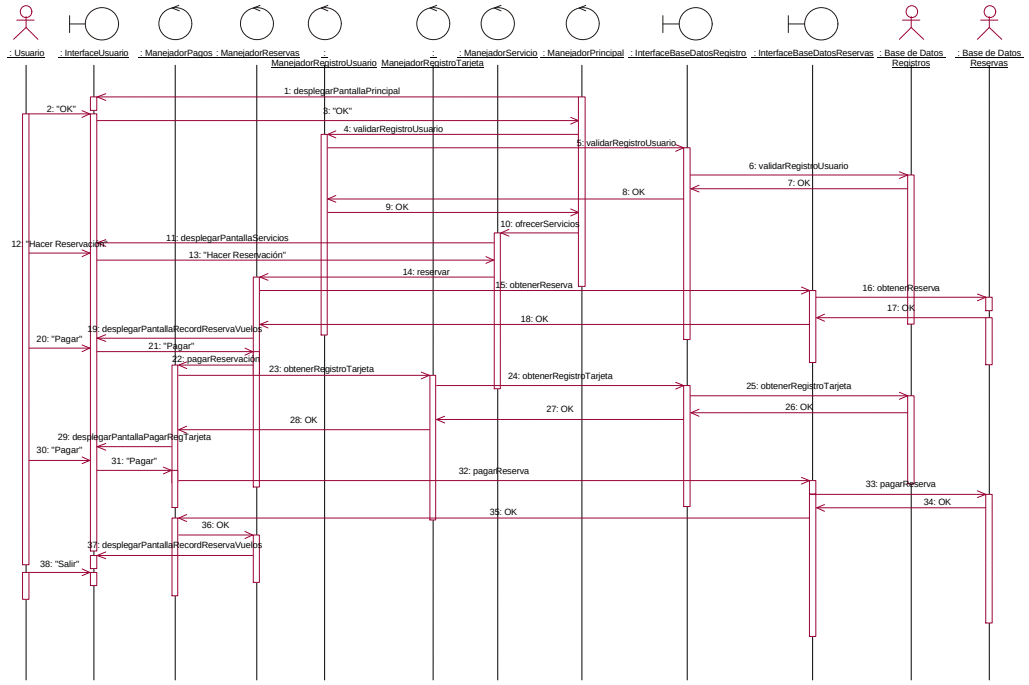

Figura 7.57 Diagrama de secuencia *Pagar Reservación* del caso de uso *Pagar Reservación.*

Reembolsar Pago

El diagrama de secuencia *Reembolsar Pago* se muestra en el diagrama 7.58. Esta secuencia es muy similar a *Pagar Reservación*, iniciando en el flujo principal de *Pagar Reservación* que incluye, por ser una extensión, la validación del usuario en *Validar Usuario* seguidos por *Ofrecer Servicios*, los subflujos *Solicitar Clave Reservación* (*S*-1), *Obtener Reservación* (*S*-3) y *Administrar Reservación* (*S*-4), del caso de uso *Hacer Reservación*. Posteriormente se ejecuta el flujo principal de *Pagar Reservación* junto con el subflujo *Reembolsar Pago* (*S*-2). Además de esto, se ejecuta el subflujo *Obtener Registro Tarjeta* (*S*-2) del caso de uso *Registrar Tarjeta*.

?? *Validar Usuario.* La secuencia comienza con la validación del usuario a partir del evento *desplegarPantallaPrincipal* de la clase *ManejadorPrincipal* a la *InterfaceUsuario*. El usuario se valida enviando el evento "*OK*". La *InterfaceUsuario* envía el mismo evento al *ManejadorPrincipal*. El *ManejadorPrincipal* envía el evento *validarRegistroUsuario* al *ManejadorRegistroUsuario*. El *ManejadorRegistroUsuario* solicita a la *InterfaceBaseDatosRegistro* que haga una validación del usuario mediante el mismo evento. La *InterfaceBaseDatosRegistro* envía a su vez el evento a la *Base de Datos Registros*, el cual contesta con un *OK* si la validación es buena. El *OK* es enviado a la *InterfaceBaseDatosRegistro*, de allí a *ManejadorRegistroUsuario* y luego a *ManejadorPrincipal* como respuesta a las secuencias de validación. Una vez el *ManejadorPrincipal* recibió este último *OK*, solicita al *ManejadorServicio* que entre en acción mediante el evento *ofrecerServicios.*

?? *Ofrecer Servicios*. A continuación podemos pasar al caso de uso *Ofrecer Servicios* donde el *ManejadorServicio* solicita entonces a la *InterfaceUsuario* el desplegado de la pantalla correspondiente mediante *desplegarPantallaServicio*. La *InterfaceUsuario* despliega esta pantalla. Para continuar con la lógica principal de este subflujo, el usuario debe presionar "Hacer Reservación" y debe incluir un número correspondiente a la clave de reservación. Se envía el mismo evento de la *InterfaceUsuario* al *ManejadorServicio*. El *ManejadorServicio* envía el evento *reservar* al *ManejadorReservas*.

?? *Hacer Reservación* subflujo *Solicitar Clave Reservación* (*S*-1). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaClaveReservas* a *InterfaceUsuario*. El usuario presiona el botón "Obtener", junto con la inserción de la clave de récord de reservas. La *InterfaceUsuario* envía el mismo evento de regreso a *ManejadorReservas.*

?? *Hacer Reservación* subflujo *Obtener Reservación* (*S*-3). A continuación el *ManejadorReservas* envía el evento *obtenerReserva* a la *InterfaceBaseDatosReservas*, la cual a su vez envía este evento al actor *Base de Datos Reservas*. El actor genera un *OK* si todo es correcto y se lo envía de regreso a *InterfaceBaseDatosReservas*, la cual luego lo envía a *ManejadorReservas.*

?? *Hacer Reservación* subflujo *Administrar Reservación* (*S*-4). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaRecordReservaVuelos* a *InterfaceUsuario*. Hasta aquí la secuencia es similar a la secuencia *Pagar Reservación*. En este momento el usuario presiona "Reembolsar". La *InterfaceUsuario* envía el mismo evento de regreso a *ManejadorReservas* el cual envía el evento *reembolsarReservación* al *ManejadorPagos.*

?? *Pagar Reservación* flujo principal. A continuación el *ManejadorPagos* envía el evento *obtenerRegistroTarjeta* al *ManejadorRegistroTarjeta*.

?? *Registrar Tarjeta* subflujo *Obtener Registro Tarjeta* (*S*-2). A continuación el *ManejadorRegistroTarjeta* envía el evento *obtenerRegTarjeta* a la *InterfaceBaseDatosRegistro* la cual envía el mismo evento al actor *Base de Datos Registros*. Este actor regresa un *OK* a la *InterfaceBaseDatosRegistro* la cual a su vez lo envía de regreso a *ManejadorRegistroTarjeta*.

?? *Pagar Reservación* flujo principal (continuación anterior). A continuación el *ManejadorRegistroTarjeta* envía el *OK* a *ManejadorPagos.*

?? *Pagar Reservación* subflujo *Reembolsar Pagos* (*S*-2). A continuación el *ManejadorPagos* envía el evento *desplegarPantallaReembolsarRegTarjetas* a la *InterfaceUsuario*. El usuario genera el evento "*Reembolsar*". La *InterfaceUsuario* recibe la petición y envía el mismo evento al *ManejadorPagos*. Este a su vez envía el evento *reembolsarReserva* a la *InterfaceBaseDatosReservas* para que haga la petición correspondiente al actor *Base de Datos Reservas*, el cual contesta con un *OK*. El *OK* es luego enviado por la *InterfaceBaseDatosRegistro* al *ManejadorPagos* el cual envía el *OK* al *ManejadorReservas*.

?? *Hacer Reservación* subflujo *Administrar Reservación* (*S*-4). A continuación el *ManejadorReservas* envía el evento *desplegarPantallaRecordReservaVuelos* a la *InterfaceUsuario* para desplegar los resultados del reembolso de reserva. En ese momento el *Usuario* presiona "*Salir*", dando por concluida la secuencia.

En resumen, la secuencia podrá iniciar con el caso de uso *Validar Usuario* seguido por el caso de uso *Ofrecer Servicios*, para luego continuar en el caso de uso *Hacer Reservación* con los subflujos *Solicitar Clave Reservación* (*S*-1), *Obtener Reservación* (*S*-3) y *Administrar Reservación* (*S*-4), y finalizar en el caso de uso *Pagar Reservación* con el flujo principal y el subflujo *Reembolsar Pago* (*S*-2). Se incluye también el caso de uso *Registrar TArjeta* subflujo *Obtener Registro Tarjeta* (*S*-2).

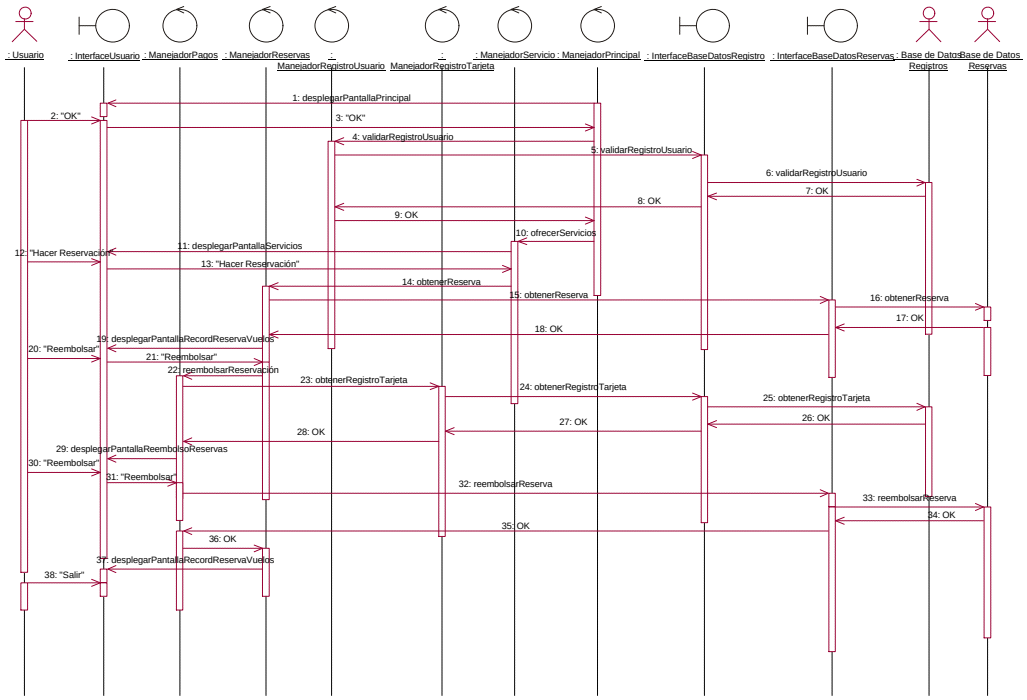

Figura 7.58 Diagrama de secuencia *Reembolsar Pago* del caso de uso *Pagar Reservación*.

7.5 Documento de Casos de Uso

A partir de las diversas secuencias analizadas y descritas en los diagramas de secuencias, podemos generar una descripción casi completa de los casos de uso del sistema de reservaciones de vuelo a nivel de análisis. Para lograr este paso, tomamos todas las descripciones y las insertamos en los flujos o subflujos correspondientes de los diversos casos de uso. Dado que las secuencias no mencionan todos los posibles eventos sino los principales, podemos en este momento completar los que sean necesarios. En particular deberemos asegurarnos que no existan discontinuidades entre las secuencias de eventos. Sin embargo, debe siempre mantenerse una buena consistencia entre los casos de uso de análisis y las secuencias anteriores que en el fondo son instanciaciones a partir de los casos de uso. Cualquier cambio en la lógica en las secuencias o casos de usuo deberá reflejarse también en los diagramas de secuencia.
En las descripciones de los casos de uso, estaremos subrayando las clases y escribiendo en letras cursivas los nombres de los eventos entre clases. Es muy importante que las frases sean claras y concisas, ya que esto facilitará posteriormente el diseño.

Validar Usuario

El flujo principal del caso de uso *Validar Usuario* se muestra a continuación.

Caso de Uso	*Validar Usuario*
Actores	*Usuario, Base de Datos Registro*
Tipo	*Inclusión*
Propósito	Validar a un usuario ya registrado para el uso del sistema de reservaciones de vuelo.
Resumen	Este caso de uso es iniciado por el *Usuario*. Valida al usuario mediante un login y password a ser validado con su respectivo registro de usuario para asi poder utilizar el sistema de reservaciones.
Precondiciones	Si el *Usuario* aún no se ha registrado, requerirá ejecutar el caso de uso *Registrar Usuario* subflujo *Crear Registro Usuario*.
Flujo Principal	El ManejadorPrincipal solicita *desplegarPantallaPrincipal* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaPrincipal. La PantallaPrincipal se *despliega*. El Usuario puede seleccionar entre las siguientes opciones: "Registrarse por Primera Vez", "OK" y "Salir". Si la actividad seleccionada es "Registrarse por Primera Vez", la PantallaPrincipal envía el evento "Registrarse por Primera Vez" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Registrarse por Primera Vez" al ManejadorPrincipal. El ManejadorPrincipal solicita *crearRegistroUsuario* al ManejadorRegistroUsuario. Se ejecuta el caso de uso *Registrar Usuario*, subflujo *Crear Registro Usuario* (*S*-1). Si la actividad seleccionada es "OK", se valida el registro de usuario mediante un *login* y un *password* insertados por el Usuario en la PantallaPrincipal. La PantallaPrincipal envía el evento "OK" a la InterfaceUsuario. La InterfaceUsuario envía el evento "OK" al ManejadorPrincipal. El ManejadorPrincipal solicita *validarRegistroUsuario* al ManejadorRegistroUsuario. El ManejadorRegistroUsuario solicita *validarRegistroUsuario* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro solicita *validarRegistroUsuario* a la Base de Datos Registro. La Base de Datos Registro valida al usuario y devuelve el *OK* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro devuelve el *OK* al ManejadorRegistroUsuario. El ManejadorRegistroUsuario devuelve el *OK* al ManejadorPrincipal. Una vez validado el usuario (*E*-1), el ManejadorPrincipal solicita *ofrecerServicio* al ManejadorServicio. Se continúa con el caso de uso *Ofrecer Servicios*. Si la actividad seleccionada es "Salir", la PantallaPrincipal envía el evento "Salir" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Salir" al ManejadorPrincipal. El ManejadorPrincipal *sale* del sistema.
Subflujos	Ninguno
Excepciones	*E-1 no hubo validación*: El *login*/*password* no se validó correctamente. Se le pide al usuario que vuelva a intentar hasta tres veces después de lo cual se saldrá del sistema.

Ofrecer Servicios

El flujo principal del caso de uso *Ofrecer Servicios* se muestra a continuación.

Caso de Uso	*Ofrecer Servicios*
Actores	*Usuario*
Tipo	*Inclusión*
Propósito	Ofrecer los diversos servicios a un usuario ya registrado para el uso del sistema de reservaciones de vuelo.
Resumen	Este caso de uso es iniciado por el *Usuario*. Tiene opciones para utilizar las diversos servicios del sistema de reservaciones.
Precondiciones	Se requiere la validación correcta del usuario.
Flujo Principal	El ManejadorServicio solicita *desplegarPantallaServicio* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaServicio. La PantallaServicio se *despliega*. El Usuario puede seleccionar entre las siguientes actividades: "Consultar Información", "Hacer Reservación", "Obtener Registro" y "Salir". Si la actividad seleccionada es "Consultar Información", la PantallaServicio envía el evento "Consultar Información" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Consultar Información" al ManejadorServicio. El ManejadorServicio solicita *consultar* al ManejadorConsultas. Se continúa con el caso de uso *Consultar Información*, subflujo *Consultar* (*S*-1). Si la actividad seleccionada es "Hacer Reservación", la PantallaServicio envía el evento "Hacer Reservación" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Hacer Reservación" al ManejadorServicio. El ManejadorServicio solicita *reservar* al ManejadorReservas. Se continúa con el caso de uso *Hacer Reservación*, subflujo *Solicitar Clave Reservación* (*S*-1). Si la actividad seleccionada es "Obtener Registro", la PantallaServicio envía el evento "Obtener Registro" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Obtener Registro" al ManejadorServicio. El ManejadorServicio solicita *registrar* al ManejadorRegistroUsuario. Se continúa con el caso de uso *Registrar Usuario*, subflujo *Obtener Registro Usuario* (*S*-2). Si la actividad seleccionada es "Salir", la PantallaServicio envía el evento "Salir" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Salir" al ManejadorServicio. El ManejadorServicio *sale* del sistema.
Subflujos	Ninguno.
Excepciones	Ninguno.

Registrar Usuario

El flujo principal del caso de uso *Registrar Usuario* se muestra a continuación.

Caso de Uso	*Registrar Usuario*
Actores	*Usuario, Base de Datos Registros*
Tipo	*Básico*
Propósito	Permitir a un usuario registrarse con el sistema de reservaciones de vuelo para su uso posterior.
Resumen	Este caso de uso es iniciado por el *Usuario*. Ofrece funcionalidad para crear, modificar y eliminar el registro de usuario con el sistema de reservaciones.
Precondiciones	Todos los subflujos, con excepción de *Registrarse Por Primera Vez*, requieren ejecutar inicialmente el caso de uso *Validar Usuario*.
Flujo Principal	Se ejecuta el caso de uso *Validar Usuario*. Dependiendo de las opciones seleccionadas por el Usuario, se continuará con los diversos subflujos de este caso de uso.

El subflujo *Crear Registro Usuario* (*S*-1) se muestra a continuación.

Subflujos	*S*-1 *Crear Registro Usuario* El ManejadorRegistroUsuario solicita *desplegarPantallaCrearRegUsuario* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaCrearRegUsuario. La PantallaCrearRegUsuario se *despliega*. Esta pantalla contiene información de registro que debe ser llenada por el Usuario, lo cual incluye nombre, apellido, calle, colonia, ciudad, país, código postal, teléfonos de la casa y oficina, número de fax, *login*, email, *password* y una entrada adicional de repetir *password* para asegurarse de su corrección. El *login* y la *password* serán utilizados por el sistema para validar al usuario. El Usuario puede seleccionar entre las siguientes actividades: "Registrar" y "Salir". Si el Usuario selecciona "Registrar", la PantallaCrearRegUsuario envía el evento "Registrar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Registrar" al ManejadorRegistroUsuario. El ManejadorRegistroUsuario solicita *crearRegistroUsuario* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro solicita *crearRegistroUsuario* a la Base de Datos Registro (*E*-1, *E*-2, *E*-3, *E*-4). La Base de Datos Registro devuelve el *OK* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro devuelve el *OK* al ManejadorRegistroUsuario. Se continúa con el subflujo *Administrar Registro Usuario* (*S*-3). Si la actividad seleccionada es "Salir", la PantallaCrearRegUsuario envía el evento "Salir" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Salir" al ManejadorRegistroUsuario. El ManejadorRegistroUsuario *sale* del sistema. (Si aún no se ha presionado "Registrar", la información será perdida).

El subflujo *Obtener Registro Usuario* (*S*-2) se muestra a continuación.

Subflujos (cont)	*S*-2 *Obtener Registro Usuario* El ManejadorRegistroUsuario solicita *obtenerRegistroUsuario* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro solicita *obtenerRegistroUsuario* a la Base de Datos Registro. La Base de Datos Registro devuelve el *OK* y el RegistroUsuario a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro devuelve el *OK* y el RegistroUsuario al ManejadorRegistroUsuario. Se continúa con el subflujo *Administrar Registro Usuario* (*S*-3).

El subflujo *Administrar Registro Usuario* (*S*-3) se muestra a continuación.

Subflujos (cont)	*S*-3 *Administrar Registro Usuario* El ManejadorRegistroUsuario solicita *desplegarPantallaObtenerRegUsuario* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaObtenerRegUsuario. La PantallaObtenerRegUsuario se *despliega*. El Usuario puede seleccionar entra las siguientes actividades: "Eliminar", "Actualizar", "Registrar Tarjeta", "Servicios" y "Salir". Si el usuario presiona "Actualizar" se ejecuta el subflujo *Actualizar Registro Usuario* (*S*-4). Si el usuario selecciona "Eliminar" se ejecuta el subflujo *Eliminar Registro Usuario* (*S*-5). Si el usuario presiona "Registrar Tarjeta", la PantallaObtenerRegUsuario envía el evento "Registrar Tarjeta" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Registrar Tarjeta" al ManejadorRegistroUsuario. El ManejadorRegistroUsuario solicita *registrarTarjeta* al ManejadorRegistroTarjeta, se continúa con el caso de uso *Registrar Tarjeta*. Si la actividad seleccionada es "Servicios", la PantallaObtenerRegUsuario envía el evento "Servicios" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Servicios" al ManejadorRegistroUsuario. El ManejadorRegistroUsuario solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si la actividad seleccionada es "Salir", la PantallaObtenerRegUsuario envía el evento "Salir" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Salir" al ManejadorRegistroUsuario. El ManejadorRegistroUsuario *sale* del sistema. (Si aún no se ha presionado "Actualizar", la nueva información será perdida).

El subflujo *Actualizar Registro Usuario* (*S*-4) se muestra a continuación.

Subflujos (cont)	*S*-4 *Actualizar Registro Usuario* La PantallaObtenerRegUsuario envía el evento " Actualizar" a la InterfaceUsuario. La InterfaceUsuario envía el evento " Actualizar" al ManejadorRegistroUsuario. El ManejadorRegistroUsuario solicita *actualizarRegistroUsuario* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro solicita *actualizarRegistroUsuario* a la Base de Datos Registro. La Base de Datos Registro actualiza el RegistroUsuario (*E*-1, *E*-3, *E*-4) y devuelve el *OK* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro devuelve el *OK* al ManejadorRegistroUsuario. Se continua con el subflujo *Administrar Registro Usuario* (*S*-3).

El subflujo *Eliminar Registro Usuario* (*S*-5) se muestra a continuación.

Subflujos (cont)	*S*-5 *Eliminar Registro Usuario* La PantallaObtenerRegUsuario envía el evento "Eliminar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Eliminar" al ManejadorRegistroUsuario. El ManejadorRegistroUsuario solicita *eliminarRegistroUsuario* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro envía el evento *eliminarRegistroUsuario* a la Base de Datos Registro. La Base de Datos Registro elimina el RegistroUsuario y devuelve el *OK* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro devuelve el *OK* al ManejadorRegistroUsuario. Se continúa con el subflujo *Crear Registro Usuario* (*S*-1).

Las excepciones del caso de uso se muestran a continuación.

Excepciones	*E-1 información incompleta*: Falta llenar información en el registro de usuario. Se le vuelve a pedir al usuario que complete el registro. *E-2 registro ya existe*: Si ya existe un registro bajo ese *login*, se le pedirá al usuario que lo cambie o que termine el caso de uso. *E-3 login incorrecto*: El *login* no es válido. Se le vuelve a pedir al usuario que complete el registro. *E-4 contraseña incorrecta*: La contraseña escogida es muy sencilla o no se validó correctamente. Se le vuelve a pedir al usuario que complete el registro.

Registrar Tarjeta

El flujo principal del caso de uso *Registrar Tarjeta* se muestra a continuación.

Caso de Uso	*Registrar Tarjeta*
Actores	*Usuario, Base de Datos Registros*
Tipo	*Extensión*
Propósito	Permitir a un usuario registrar una tarjeta de créditos con el sistema de reservaciones de vuelo para poder pagar boletos.
Resumen	Este caso de uso es iniciado por el *Usuario*. Ofrece funcionalidad para para crear, modificar y eliminar el registro de tarjeta usuario para poder pagar las reservaciones directamente con el sistema de reservaciones.
Precondiciones	El usuario ya se debe haberse registrado mediante la activación del caso de uso *Registrar Usuario*.
Flujo Principal	Se continúa con el subflujo *Obtener Registro Tarjeta* (*S*-2). Si no existe un RegistroTarjeta válido se continúa con el subflujo *Crear Registro Tarjeta* (*S*-1). De lo contrario, si ya existe un RegistroTarjeta válido, se continúa con el subflujo *Administrar Registro Tarjeta* (*S*-3).

El subflujo *Crear Registro Tarjeta* (*S*-1) se muestra a continuación.

Subflujos	*S*-1 *Crear Registro Tarjeta* El ManejadorRegistroTarjeta solicita *desplegarPantallaCrearRegTarjeta* a la InterfaceUsuario. La InterfaceUsuario *despliega* a la PantallaCrearRegTarjeta. La PantallaCrearRegTarjeta se *despliega*. Esta pantalla contiene información que debe ser llenada por el Usuario, lo cual incluye el nombre como aparece el la tarjeta, número de tarjeta, el tipo de tarjeta, y la fecha de vencimiento. El Usuario puede seleccionar entre las siguientes actividades: "Registrar", "Servicios" y "Salir". Si el Usuario selecciona "Registrar", la PantallaCrearRegTarjeta envía el evento "Registrar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Registrar" al ManejadorRegistroTarjeta. El ManejadorRegistroTarjeta solicita *crearRegistroTarjeta* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro solicita *crearRegistroTarjeta* a la Base de Datos Registro (*E*-1). La Base de Datos Registro devuelve el *OK* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro devuelve el *OK* al ManejadorRegistroTarjeta. Se continúa con el subflujo *Administrar Registro Tarjeta* (*S*-3). Si la actividad seleccionada es "Servicios", la PantallaCrearRegTarjeta envía el evento "Servicios" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Servicios" al ManejadorRegistroTarjeta. El ManejadorRegistroTarjeta solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si la actividad seleccionada es "Salir", la PantallaCrearRegTarjeta envía el evento "Salir" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Salir" al ManejadorRegistroTarjeta. El ManejadorRegistroTarjeta *sale* del sistema. (Si aún no se ha presionado "Registrar", la información ésta será perdida).

El subflujo *Obtener Registro Tarjeta* (*S*-2) se muestra a continuación.

Subflujos (cont)	*S*-2 *Obtener Registro Tarjeta* El ManejadorRegistroTarjeta solicita *obtenerRegistroTarjeta* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro solicita *obtenerRegistroTarjeta* a la Base de Datos Registro. La Base de Datos Registro devuelve el *OK* y el RegistroTarjeta a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro devuelve el *OK* y el RegistroTarjeta al ManejadorRegistroTarjeta. Se regresa al flujo anterior.

El subflujo *Administrar Registro Tarjeta* (*S*-3) se muestra a continuación.

Subflujos (cont)	*S*-3 *Administrar Registro Tarjeta* El ManejadorRegistroTarjeta solicita *desplegarPantallaObtenerRegTarjeta* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaObtenerRegTarjeta. La PantallaObtenerRegTarjeta se *despliega*. El Usuario puede seleccionar entre las siguientes actividades: "Actualizar", "Eliminar", "Servicios" y "Salir". Si el usuario presiona "Actualizar" se ejecuta el subflujo *Actualizar Registro Tarjeta* (*S*-4). Si el usuario presiona "Eliminar" se ejecuta el subflujo *Eliminar Registro Tarjeta* (*S*-5). Si la actividad seleccionada es "Servicios", la PantallaObtenerRegTarjeta envía el evento "Servicios" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Servicios" al ManejadorRegistroTarjeta. El ManejadorRegistroTarjeta solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si la actividad seleccionada es "Salir", la PantallaObtenerRegTarjeta envía el evento "Salir" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Salir" al ManejadorRegistroTarjeta. El ManejadorRegistroTarjeta *sale* del sistema. (Si aún no se ha presionado "Actualizar", la nueva información ésta será perdida).

El subflujo *Actualizar Registro Tarjeta* (*S*-4) se muestra a continuación.

Subflujos (cont)	*S*-4 *Actualizar Registro Tarjeta* La PantallaObtenerRegTarjeta envía el evento "Actualizar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Actualizar" al ManejadorRegistroTarjeta. El ManejadorRegistroTarjeta solicita *actualizarRegistroTarjeta* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro solicita *actualizarRegistroTarjeta* a la Base de Datos Registro (*E*-1). La Base de Datos Registro actualiza el RegistroTarjeta y devuelve el evento *OK* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro devuelve el *OK* al ManejadorRegistroTarjeta. Se continua con el subflujo *Administrar Registro Tarjeta* (*S*-3).

El subflujo *Eliminar Registro Tarjeta* (*S*-5) se muestra a continuación.

Subflujos (cont)	*S*-5 *Eliminar Registro Tarjeta* La PantallaObtenerRegTarjeta envía el evento "Eliminar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Eliminar" al ManejadorRegistroTarjeta. El ManejadorRegistroTarjeta solicita *eliminarRegistroTarjeta* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro solicita *eliminarRegistroTarjeta* a la Base de Datos Registro. La Base de Datos Registro elimina el RegistroTarjeta (*E*-1) y devuelve el *OK* a la InterfaceBaseDatosRegistro. La InterfaceBaseDatosRegistro devuelve el *OK* al ManejadorRegistroTarjeta. Se continua con el subflujo *Crear Registro Tarjeta* (*S*-1).

Las excepciones del caso de uso se muestran a continuación.

Excepciones	*E-1 información incompleta*: Falta llenar información indispensable para completar el registro de tarjeta. Se le vuelve a pedir al usuario que complete el registro de tarjeta.

Consultar Información

El flujo principal del caso de uso *Consultar Información* se muestra a continuación.

Caso de Uso	*Consultar Información*
Actores	*Usuario, Base de Datos Reservas*
Tipo	*Básico*
Propósito	Permitir a un usuario consultar información con el sistema de reservaciones de vuelo.
Resumen	Este caso de uso es iniciado por el *Usuario*. Ofrece funcionalidad para consultar información de horarios, tarifas y estado de vuelos con el sistema de reservaciones.
Precondiciones	Se requieren haber ejecutado anteriormente el caso de uso *Validar Usuario*.
Flujo Principal	Se ejecuta el caso de uso Validar Usuario. Dependiendo de las opciones seleccionadas por el Usuario, se continuará con los diversos subflujos de este caso de uso.

El subflujo *Consultar* (*S*-1) se muestra a continuación.

Subflujos	*S*-1 *Consultar* El ManejadorConsultas solicita *desplegarPantallaConsultas* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaConsultas. La PantallaConsultas se *despliega*. El Usuario puede seleccionar de las siguientes actividades: "Horarios", "Tarifas", "Estado", “Servicios” y “Salir”. Si la actividad seleccionada es “Horarios”, la PantallaConsultas envía el evento “Horarios” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Horarios” al ManejadorConsultas. El ManejadorConsultas solicita *consultarHorarios* al ManejadorConsultasHorarios. Se continúa con el subflujo *Consultar Horarios* (*S*-2). Si el usuario presiona “Tarifas”, la PantallaConsultas envía el evento “Tarifas” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Tarifas” al ManejadorConsultas. El ManejadorConsultas solicita *consultarTarifas* al ManejadorConsultasTarifas. Se continúa con el subflujo *Consultar Tarifas* (*S*-4). Si el usuario presiona “Estado”, la PantallaConsultas envía el evento “Estado” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Estado” al ManejadorConsultas. El ManejadorConsultas solicita *consultarEstado* al ManejadorConsultasEstado. Se continúa con el subflujo *Consultar Estado* (*S*-6). Si el usuario presiona “Servicios”, la PantallaConsultas envía el evento “Servicios” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Servicios” al ManejadorConsultas. El ManejadorConsultas solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si el usuario presiona "Salir", la PantallaConsultas envía el evento “Salir” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Salir” al ManejadorConsultas. El ManejadorConsultas *sale* del sistema.

El subflujo *Consultar Horarios* (*S*-2) se muestra a continuación.

Subflujos (cont)	*S*-2 *Consultar Horarios* El ManejadorConsultaHorarios solicita *desplegarPantallaConsultaHorarios* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaConsultaHorarios. La PantallaConsultaHorarios se *despliega*. Esta pantalla debe ser llenada con información de ciudad de origen y destino, y preferencias opcionales de: aerolínea, horario y una opción de vuelo sólo directo. El Usuario puede seleccionar de las siguientes actividades: "Consultar", "Servicios" y "Salir". Si el usuario presiona “Consultar”, la PantallaConsultaHorarios envía el evento “Consultar” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Consultar” al ManejadorConsultaHorarios. El ManejadorConsultaHorarios solicita *consultarHorarios* a la InterfaceBaseDatosReservas (*E*-1, *E*-2). La InterfaceBaseDatosReservas solicita *consultarHorarios* a la Base de Datos Reservas. La Base de Datos Reservas devuelve los Vuelos a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas devuelve los Vuelos al ManejadorConsultaHorarios. Se continúa con el subflujo *Devolver Horarios* (*S*-3). Si el usuario presiona “Servicios”, la PantallaConsultaHorarios envía el evento “Servicios” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Servicios” al ManejadorConsultaHorarios. El ManejadorConsultaHorarios solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si el usuario presiona "Salir", la PantallaConsultaHorarios envía el evento “Salir” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Salir” al ManejadorConsultaHorarios. El ManejadorConsultaHorarios *sale* del sistema..

El subflujo *Devolver Horarios* (*S*-3) se muestra a continuación.

Subflujos (cont)	*S*-3 *Devolver Horarios* El ManejadorConsultaHorarios solicita *desplegarPantallaResultadoHorarios* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaResultadoHorarios. La PantallaResultadoHorarios *despliega* información de Vuelo, Aerolínea, Horarios de Salida y Llegada y Aeropuertos de Origen, Destino y Escalas con posibles conexiones. El usuario puede seleccionar entre las siguientes opciones: “+”, "-", “Nueva Consulta”, "Servicios" y "Salir". Si el usuario presiona "+", la PantallaResultadoHorarios envía el evento “+” a la InterfaceUsuario. La InterfaceUsuario envía el evento “+” al ManejadorConsultaHorarios. Se continúa al inicio de este subflujo (presentando resultados adicionales). Si el usuario presiona "-", la PantallaResultadoHorarios envía el evento “-” a la InterfaceUsuario. La InterfaceUsuario envía el evento “-” al ManejadorConsultaHorarios. Se continúa al inicio de este subflujo (presentando resultados anteriores). Si el usuario presiona “Nueva Consulta”, la PantallaResultadoHorarios envía el evento “Nueva Consulta” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Nueva Consulta” al ManejadorConsultaHorarios. Se continúa con el subflujo *Consultar Horarios* (*S*-2). Si la actividad seleccionada es "Servicios", la PantallaResultadoHorarios envía el evento “Servicios” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Servicios” al ManejadorConsultaHorarios. El ManejadorConsultaHorarios solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si el usuario presiona "Salir”, la PantallaResultadoHorarios envía el evento “Salir” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Salir” al ManejadorConsultaHorarios. El ManejadorConsultaHorarios *sale* del sistema.

El subflujo *Consultar Tarifas* (*S*-4) se muestra a continuación.

Subflujos (cont)	*S*-4 *Consultar Tarifas* El ManejadorConsultaTarifas solicita *desplegarPantallaConsultaTarifas* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaConsultaTarifas. La PantallaConsultaTarifas se *despliega*. Esta pantalla debe ser llenada con información de ciudad de origen y destino, y preferencias opcionales: fecha de salida, fecha de regreso, aerolínea, clase, y las opciones de organizar la información por menor tarifa, una opción de vuelo si sólo directo, y si la tarifa se basa en viaje redondo. El Usuario puede seleccionar de las siguientes actividades: "Consultar", "Servicios" y "Salir". Si el usuario presiona “Consultar”, la PantallaConsultaTarifas envía el evento “Consultar” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Consultar” al ManejadorConsultaTarifas. El ManejadorConsultaTarifas solicita *consultarTarifas* a la InterfaceBaseDatosReservas (*E*-1,*E*-2). La InterfaceBaseDatosReservas solicita *consultarTarifas* a la Base de Datos Reservas. La Base de Datos Reservas devuelve los Vuelos a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas devuelve los Vuelos al ManejadorConsultaTarifas. Se continúa con el subflujo *Devolver Tarifas* (*S*-5). Si el usuario presiona “Servicios”, la PantallaConsultaTarifas envía el evento “Servicios” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Servicios” al ManejadorConsultaTarifas. El ManejadorConsultaTarifas solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si el usuario presiona "Salir", la PantallaConsultaTarifas envía el evento “Salir” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Salir” al ManejadorConsultaTarifas. El ManejadorConsultaTarifas *sale* del sistema.

El subflujo *Devolver Tarifas* (*S*-5) se muestra a continuación.

Subflujos (cont)	*S*-5 *Devolver Tarifas* El ManejadorConsultaTarifas solicita *desplegarPantallaResultadoTarifas* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaResultadoTarifas. La PantallaResultadoTarifas *despliega* información de Vuelo, Aerolínea, Horarios de Salida y Llegada, Aeropuertos de Origen, Destino y Escalas con posibles conexiones y Tarifas de ida y vuelta. El usuario puede seleccionar entre las siguientes opciones: “+”, "-", "Nueva Consulta", "Servicios" y "Salir". Si el usuario presiona "+", la PantallaResultadoTarifas envía el evento “+” a la InterfaceUsuario. La InterfaceUsuario envía el evento “+” al ManejadorConsultaTarifas. Se continúa al inicio de este subflujo (presentando resultados adicionales). Si el usuario presiona "-", la PantallaResultadoTarifas envía el evento “-” a la InterfaceUsuario. La InterfaceUsuario envía el evento “-” al ManejadorConsultaTarifas. Se continúa al inicio de este subflujo (presentando resultados anteriores). Si el usuario presiona “Nueva Consulta”, la PantallaResultadoTarifas envía el evento “Nueva Consulta” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Nueva Consulta” al ManejadorConsultaTarifas. Se continúa con el subflujo *Consultar Tarifas* (*S*-4). Si la actividad seleccionada es "Servicios", la PantallaResultadoTarifas envía el evento “Servicios” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Servicios” al ManejadorConsultaTarifas. El ManejadorConsultaTarifas solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si el usuario presiona "Salir”, la PantallaResultadoTarifas envía el evento “Salir” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Salir” al ManejadorConsultaTarifas. El ManejadorConsultaTarifas *sale* del sistema.

El subflujo *Consultar Estado* (*S*-6) se muestra a continuación.

Subflujos (cont)	*S*-6 *Consultar Estado* El ManejadorConsultaEstado solicita *desplegarPantallaConsultaEstado* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaConsultaEstado. La PantallaConsultaEstado se *despliega*. Esta pantalla debe ser llenada con información de ciudad de origen y destino, la aerolínea, el número de vuelo, y la opción de vuelo de hoy. El Usuario puede seleccionar de las siguientes actividades: "Consultar", "Regresar" y "Salir". Si el usuario presiona “Consultar”, la PantallaConsultaEstado envía el evento “Consultar” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Consultar” al ManejadorConsultaEstado. El ManejadorConsultaEstado solicita *consultarEstado* a la InterfaceBaseDatosReservas (*E*-1,*E*-2). La InterfaceBaseDatosReservas solicita *consultarEstado* a la Base de Datos Reservas. La Base de Datos Reservas devuelve el Vuelo a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas devuelve el Vuelo al ManejadorConsultaEstado. Se continúa con el subflujo *Devolver Estado* (*S*-7). Si el usuario presiona “Servicios”, la PantallaConsultaEstado envía el evento “Servicios” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Servicios” al ManejadorConsultaEstado. El ManejadorConsultaEstado solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si el usuario presiona "Salir", la PantallaConsultaEstado envía el evento “Salir” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Salir” al ManejadorConsultaEstado. El ManejadorConsultaEstado *sale* del sistema.

El subflujo *Devolver Estado* (*S*-7) se muestra a continuación.

Subflujos (cont)	*S*-7 *Devolver Estado* El ManejadorConsultaEstado solicita *desplegarPantallaResultadoEstado* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaResultadoEstado. La PantallaResultadoEstado *despliega* la información de Vuelo, Aerolínea, Horarios de Salida y Llegada, Aeropuertos de Origen, Destino y Escalas con posibles conexiones y Avión utilizado. El usuario puede seleccionar entre las siguientes opciones: "Nueva Consulta", "Servicios" y "Salir". Si el usuario presiona "Nueva Consulta", la PantallaResultadoEstado envía el evento "Nueva Consulta" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Nueva Consulta" al ManejadorConsultaEstado. Se continúa con el subflujo *Consultar Estado* (*S*-6). Si la actividad seleccionada es "Servicios", la PantallaResultadoEstado envía el evento "Servicios" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Servicios" al ManejadorConsultaEstado. El ManejadorConsultaEstado solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si el usuario presiona "Salir", la PantallaResultadoEstado envía el evento "Salir" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Salir" al ManejadorConsultaEstado. El ManejadorConsultaEstado *sale* del sistema..

Las excepciones del caso de uso se muestran a continuación.

Excepciones	*E-1 información incompleta*: Falta llenar información indispensable, ciudad de origen o de destino. Se le vuelve a pedir al usuario la información. *E-2 información inválida*: Una de las entradas de la solicitud es incorrecta.

Hacer Reservación

El flujo principal del caso de uso *Hacer Reservación* se muestra a continuación.

Caso de Uso	*Hacer Reservación*
Actores	*Usuario, Base de Datos Reservas*
Tipo	*Básico*
Propósito	Permitir a un usuario hacer reservaciones con el sistema de reservaciones de vuelo.
Resumen	Este caso de uso es iniciado por el *Usuario*. Ofrece funcionalidad para crear, obtener, modificar y eliminar reservaciones de vuelos con el sistema de reservaciones.
Precondiciones	Se debe haber ejecutado anteriormente el caso de uso Validar Usuario.
Flujo Principal	Se ejecuta el caso de uso *Validar Usuario*. Dependiendo de las opciones seleccionadas por el Usuario, se continuará con los diversos subflujos de este caso de uso.

El subflujo *Solicitar Clave Reservación* (*S*-1) se muestra a continuación.

Subflujos	*S*-1 *Solicitar Clave Reservación* El ManejadorReservas solicita *desplegarPantallaClaveReservas* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaClaveReservas. La PantallaClaveReservas se *despliega*. El *Usuario* puede seleccionar entre las siguientes opciones: "Crear", "Obtener", "Servicios" y "Salir". Si el Usuario selecciona "Crear", la PantallaClaveReservas envía el evento "Crear" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Crear" al ManejadorReservas. Se continúa con el subflujo *Crear Reservación* (*S*-2). Si el Usuario selecciona "Obtener", la PantallaClaveReservas envía el evento "Obtener" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Obtener" al ManejadorReservas. Se continúa con el subflujo *Obtener Reservación* (*S*-3) (*E*-1). Si el usuario presiona "Servicios", la PantallaClaveReservas envía el evento "Servicios" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Servicios" al ManejadorReservas. El ManejadorReservas solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si el usuario presiona "Salir", la PantallaClaveReservas envía el evento "Salir" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Salir" al ManejadorReservas. El ManejadorReservas *sale* del sistema.

El subflujo *Crear Reservación* (*S*-2) se muestra a continuación.

Subflujos (cont)	*S*-2 *Crear Reservación* El ManejadorReservas solicita *desplegarPantallaCrearReservaVuelos* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaCrearReservaVuelos. La PantallaCrearReservaVuelos se *despliega*. Esta pantalla debe ser llenada con información de apellido y nombre del pasajero, un número de viajero frecuente opcional, aerolínea, número de vuelo, ciudad de origen y destino, fecha, clase, una opción de solicitar asiento y si desea ventana o pasillo, y opcionalmente comida vegetal o carne. El Usuario puede seleccionar entre las siguientes actividades: "Agregar", "Borrar", "+", "-", "Reservar", "Servicios" y "Salir". Si el usuario presiona "Agregar", la PantallaCrearReservaVuelos envía el evento "Agregar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Agregar" al ManejadorReservas. Se continúa al inicio de este subflujo (solicitando reservaciones adicionales). Si el usuario presiona "Borrar", la PantallaCrearReservaVuelos envía el evento "Borrar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Borrar" al ManejadorReservas. Se borran los datos recién insertados y se continúa al inicio de este subflujo (solicitando reservaciones adicionales). Si el usuario presiona "+", la PantallaCrearReservaVuelos envía el evento "+" a la InterfaceUsuario. La InterfaceUsuario envía el evento "+" al ManejadorReservas. Se continúa al inicio de este subflujo (presentando solicitudes adicionales). Si el usuario presiona "-", la PantallaCrearReservaVuelos envía el evento "-" a la InterfaceUsuario. La InterfaceUsuario envía el evento "-" al ManejadorReservas. Se continúa al inicio de este subflujo (presentando solicitudes anteriores). Si el usuario selecciona "Reservar", la PantallaCrearReservaVuelos envía el evento "Reservar"a la InterfaceUsuario. La InterfaceUsuario envía el evento "Reservar"al ManejadorReservas (*E*-2,*E*-3). El ManejadorReservas solicita *crearReserva* a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas solicita *crearReserva* a la Base de Datos Reservas. La Base de Datos Reservas devuelve la Reservación a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas devuelve la Reservación al ManejadorReservas (*E*-4). Se continua con el subflujo *Administrar Reservación* (*S*-4). Si la actividad seleccionada es "Servicios", la PantallaCrearReservaVuelos envía el evento "Servicios" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Servicios" al ManejadorReservas. El ManejadorReservas solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si la actividad seleccionada es "Salir", la PantallaCrearReservaVuelos envía el evento "Salir" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Salir" al ManejadorReservas. El ManejadorReservas *sale* del sistema.

El subflujo *Obtener Reservación* (*S*-3) se muestra a continuación.

Subflujos (cont)	*S*-3 *Obtener Reservación* El ManejadorReservas solicita *obtenerReserva* a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas solicita *obtenerReserva* a la Base de Datos Reservas (*E*-1). La Base de Datos Reservas devuelve la Reservación a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas devuelve la Reservación al ManejadorReserva. Se continúa con el subflujo *Administrar Reservación* (*S*-4).

El subflujo *Administrar Reservación* (*S*-4) se muestra a continuación.

Subflujos (cont)	*S*-4 *Administrar Reservación* El ManejadorReserva solicita *desplegarPantallaRecordReservaVuelos* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaRecordReservaVuelos. La PantallaRecordReservaVuelos *despliega* información de Reservación, Vuelo, Aerolínea, Horarios de Salida y Llegada, Aeropuertos de Origen, Destino y Escalas con posibles conexiones y Tarifas de ida y vuelta. El Usuario pueden escoger las siguientes opciones: "Actualizar", "Eliminar", "+", "-", "Nueva Reserva", "Pagar", "Reembolsar", "Servicios" y "Salir". Si el usuario presiona "Actualizar" se ejecuta el subflujo *Actualizar Reservación* (*S*-5). Si el usuario presiona "Eliminar" se ejecuta el subflujo *Eliminar Reservación* (*S*-6). Si el usuario presiona "+", la PantallaRecordReservaVuelos envía el evento "+" a la InterfaceUsuario. La InterfaceUsuario envía el evento "+" al ManejadorReservas. Se continúa al inicio de este subflujo (presentando resultados adicionales). Si el usuario presiona "-", la PantallaRecordReservaVuelos envía el evento "-" a la InterfaceUsuario. La InterfaceUsuario envía el evento "-" al ManejadorReservas. Se continúa al inicio de este subflujo (presentando resultados anteriores). Si el usuario selecciona "Nueva Reserva", la PantallaRecordReservaVuelos envía el evento "Nueva Reserva" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Nueva Reserva" al ManejadorReservas. Se continúa con el subflujo *Crear Reservación* (*S*-2). Si la actividad seleccionada es "Pagar", la PantallaRecordReservaVuelos envía el evento "Pagar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Pagar" al ManejadorReservas. El ManejadorReservas solicita *pagarReservación* al ManejadorPagos, se continúa con el caso de uso *Pagar Reservación*. Si la actividad seleccionada es "Reembolsar", la PantallaRecordReservaVuelos envía el evento "Reembolsar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Reembolsar" al ManejadorReservas. El ManejadorReservas solicita *reembolsarReservación* al ManejadorPagos, se continúa con el caso de uso *Pagar Reservación* Si la actividad seleccionada es "Servicios", la PantallaRecordReservaVuelos envía el evento "Servicios" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Servicios" al ManejadorReservas. El ManejadorReservas solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si la actividad seleccionada es "Salir", la PantallaRecordReservaVuelos envía el evento "Salir" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Salir" al ManejadorReservas. El ManejadorReservas *sale* del sistema.

El subflujo *Actualizar Reservación* (*S*-5) se muestra a continuación.

Subflujos (cont)	*S*-5 *Actualizar Reservación* La PantallaRecordReservaVuelos envía el evento "Actualizar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Actualizar" al ManejadorReservas. El ManejadorReservas solicita *actualizarReserva* a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas solicita *actualizarReserva* a la Base de Datos Reservas (*E*-2,*E*-3). La Base de Datos Reservas devuelve la Reservación a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas devuelve la Reservación al ManejadorReserva (*E*-4). Se continua con el subflujo *Adminsitrar Reservación* (*S*-4).

El subflujo *Eliminar Reservación* (*S*-6) se muestra a continuación.

Subflujos (cont)	*S*-6 *Eliminar Reservación* La PantallaRecordReservaVuelos envía el evento "Eliminar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Eliminar" al ManejadorReservas. El ManejadorReservas solicita *eliminarReserva* a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas solicita *eliminarReserva* a la Base de Datos Reservas (*E*-5). La Base de Datos Reservas devuelve el *OK* a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas devuelve el *OK* al ManejadorReserva. Se continua con el subflujo *Crear Reservación* (*S*-2).

Las excepciones del caso de uso se muestran a continuación.

Excepciones	*E-1 récord inválido:* No existe el récord especificado. *E-2 información incompleta*: Falta llenar información indispensable, ciudad de origen o de destino. Se le vuelve a pedir al usuario la información. *E-3 información inválida*: Una de las entradas de la solicitud es incorrecta. *E-4 reserva sin éxito*: No se logró obtener una reserva. *E-5 eliminación reserva sin éxito*: No se logró eliminar la reserva.

Pagar Reservación

El flujo principal del caso de uso *Pagar Reservación* se muestra a continuación.

Caso de Uso	*Pagar Reservación*
Actores	*Usuario, Base de Datos Reservas*
Tipo	*Extensión*
Propósito	Permitir a un usuario pagar reservaciones con el sistema de reservaciones de vuelo.
Resumen	Este caso de uso es iniciado por el *Usuario*. Ofrece funcionalidad para pagar y reembolsar pagos de reservaciones de vuelos con el sistema de reservaciones mediante tarjetas de crédito registradas con el sistema.
Precondiciones	Se requieren haber ejecutado anteriormente el caso de uso *Validar Usuario* y tener una Reserva ya hecha mediante el caso de uso Hacer Reservación subflujo *Crear Reservación* (*S*-2)
Flujo Principal	El ManejadorPagos solicita *obtenerRegistroTarjeta* al ManejadorRegistroTarjeta. Se continúa con el caso de uso *Registrar Tarjeta*, subflujo *Obtener Registro Tarjeta* (*S*-2). El ManejadorRegistroTarjeta devuelve el OK y el RegistroTarjeta al ManejadorPagos. Dependiendo si la solicitud original del ManejadorReservas al ManejadorPagos fue *pagarReservación*, se continúa el subflujo *Pagar Reservación* (*S*-1); si fue *reembolsarReservación*, se continúa con el subflujo *Reembolsar Pago* (*S*-2).

El subflujo *Pagar Reservación* (*S*-1) se muestra a continuación.

Subflujos	*S*-1 *Pagar Reservación* El ManejadorPagos solicita *desplegarPantallaPagarRegistroTarjeta* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaPagarRegistroTarjeta. La PantallaPagarRegistroTarjeta *despliega* el RegistroTarjeta la cual incluye información de nombre como aparece en la tarjeta, número de tarjeta, el tipo de tarjeta, la fecha de vencimiento y la cantidad a pagar (*E*-1). El Usuario podrá seleccionar entre las siguientes actividades: "Pagar", "Servicios" y "Salir". Si la actividad seleccionada es “Pagar”, la PantallaRecordReservaVuelos envía el evento "Pagar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Pagar" al ManejadorPagos. El ManejadorPagos solicita *pagarReserva* a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas solicita *pagarReserva* a la Base de Datos Reservas. La Base de Datos Reservas devuelve la Reservación a la InterfaceBaseDatosReservas (*E*-2). La InterfaceBaseDatosReservas devuelve la Reservación al ManejadorPagos. El ManejadorPagos devuelve la Reservación al ManejadorReservas. Se continúa con el caso de uso *Hacer Reservación*, subflujo *Solicitar Clave Reservación* (*S*-1). Si la actividad seleccionada es "Servicios", la PantallaPagarRegistroTarjeta envía el evento “Servicios” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Servicios” al ManejadorPagos. El ManejadorPagos solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si la actividad seleccionada es "Salir", la PantallaPagarRegistroTarjeta envía el evento “Salir” a la InterfaceUsuario. La InterfaceUsuario envía el evento “Salir” al ManejadorPagos. El ManejadorPagos *sale* del sistema.

El subflujo *Reembolsar Pago* (*S*-2) se muestra a continuación.

Subflujos (cont)	*S*-2 *Reembolsar Pago* El ManejadorPagos solicita *desplegarPantallaReembolsarRegistroTarjeta* a la InterfaceUsuario. La InterfaceUsuario *despliega* la PantallaReembolsarRegistroTarjeta. La PantallaReembolsarRegistroTarjeta *despliega* la información, lo cual incluye el nombre como aparece en la tarjeta, número de tarjeta, el tipo de tarjeta, la fecha de vencimiento y la cantidad a reembolsar (*E*-1). El Usuario podrá seleccionar entre las siguientes actividades: "Reembolsar", "Servicios" y "Salir". Si la actividad seleccionada es "Reembolsar", la PantallaRecordReservaVuelos envía el evento "Reembolsar" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Reembolsar" al ManejadorPagos. El ManejadorPagos solicita *reembolsarReserva* a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas solicita *reembolsarReserva* a la Base de Datos Reservas (*E*-3, *E*-4). La Base de Datos Reservas devuelve la Reservación a la InterfaceBaseDatosReservas. La InterfaceBaseDatosReservas devuelve la Reservación al ManejadorPagos. El ManejadorPagos devuelve la Reservación al ManejadorReservas. Se continúa con el caso de uso *Hacer Reservación*, subflujo *Solicitar Clave Reservación* (*S*-1). Si la actividad seleccionada es "Servicios", la PantallaReembolsarRegistroTarjeta envía el evento "Servicios" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Servicios" al ManejadorPagos. El ManejadorPagos solicita *ofrecerServicio* al ManejadorServicio, se continúa con el caso de uso *Ofrecer Servicios*. Si la actividad seleccionada es "Salir", la PantallaReembolsarRegistroTarjeta envía el evento "Salir" a la InterfaceUsuario. La InterfaceUsuario envía el evento "Salir" al ManejadorPagos. El ManejadorPagos *sale* del sistema.

Las excepciones del caso de uso se muestran a continuación.

Excepciones	*E-1 récord inválido:* No existe el récord especificado. El usuario deberá insertar los datos de la tarjeta. *E-2 pago inválido*: El pago no tuvo éxito o la información de pago está incompleta. *E-3 pago inexistente*: La reserva no ha sido aún pagada. *E-4 devolución inválido*: No se pudo hacer la devolución del pago.

7.6 Diccionario de Clases

Como última etapa del modelo de análisis, se actualiza el diccionario de datos originalmente descrito para el dominio del problema para incluir todas las clases identificadas durante el modelo de análisis.
Aunque no es obigatorio, aprovechamos para separar estas clases en diferentes módulos para lograr una mejor organización y correspondencia entre clases y casos de uso. Aquellas clases que participan en varios casos de uso se pueden asignar a módulos adicionales, como veremos a continuación para el sistema de reservaciones de vuelo. Comenzamos con cuatro módulos o paquetes principales: *InterfaceUsuario*, *Principal*, *Registro* y *Sevicios*, como se muestra en la Figura 7.59.

Figura 7.59 Módulos principales del sistema de reservaciones de vuelo.

InterfaceUsuario

El módulo *InterfaceUsuario* está compuesto por una sóla clase:
?? **InterfaceUsuario** – Clase Borde. Toda la interacción con el usuario se hace por medio de la borde de usuario.

Principal

El módulo *Principal* está compuesto por dos clases:

?? **PantallaPrincipal** - Clase Borde. Pantalla principal (*P*-1).

?? **ManejadorPrincipal** - Clase Control. El manejador principal es el encargado de desplegar la pantalla principal de interacción con el usuario, y luego delegar las diferentes funciones a los manejadores especializados apropiados.

Registro

El módulo *Registro* se divide en los siguientes módulos: *RegistroPrincipal*, *Usuario* y *Tarjeta*, como se muestra en la Figura 7.60.

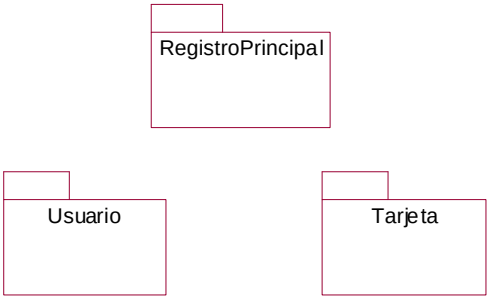

Figura 7.60 Módulos adicionales del módulo *Registro*.

RegistroPrincipal

El módulo *RegistroPrincipal* está compuesto por una sóla clase:

?? **InterfaceBaseDatosRegistro** - Clase Borde. La información de cada usuario se almacena en la base de datos de registro la cual se accesa mediante la borde de la base de datos de registro. Esto permite validar a los distintos usuarios además de guardar información sobre la tarjeta de crédito para pagos en línea.

Usuario

El módulo *Usuario* está compuesto por las clases:

?? **PantallaCrearRegUsuario** - Clase Borde. Pantalla de solicitud de registro de usuario (*P*-3).

?? **PantallaObtenerRegUsuario** - Clase Borde. Pantalla de devolución con información de registro de usuario (*P*-4).

?? **RegistroUsuario** - Clase Entidad. Para poder utilizar el sistema de reservaciones, el usuario debe estar registrado con el sistema. El registro contiene información acerca del usuario que incluye nombre, dirección, colonia, ciudad, país, código postal, teléfono de casa, teléfono de oficina, fax, email, login y password.

?? **ManejadorRegistroUsuario** - Clase Control. El manejador de registro de usuario se encarga de todo lo relacionado con registro del usuario para poder utilizar el sistema.

Tarjeta

El módulo *Tarjeta* está compuesto por las clases:

?? **PantallaCrearRegTarjeta** - Clase Borde. Pantalla de solicitud de registro de tarjeta (*P*-5).

?? **PantallaObtenerRegTarjeta** - Clase Borde. Pantalla de devolución con información de registro de tarjeta (*P*-6).

?? **RegistroTarjeta** - Clase Entidad. Para poder hacer un pago con una tarjeta de crédito, se debe tener un registro de tarjeta. El registro contiene información acerca de la tarjeta incluyendo nombre, número, expedidor y vencimiento. LA tarjeta está ligada a un registro de usuario.

?? **ManejadorRegistroTarjeta** - Clase Control. El manejador de registro de tarjeta se encarga de todo lo relacionado con registro de la tarjeta del usuario para poder pagar las reservaciones.

Servicios

El módulo *Servicio* se divide en los siguientes módulos: *ServicioPrincipal*, *Dominio*, *Consultas*, *Reservas*, y *Pagos*, como se muestra en la Figura 7.61.

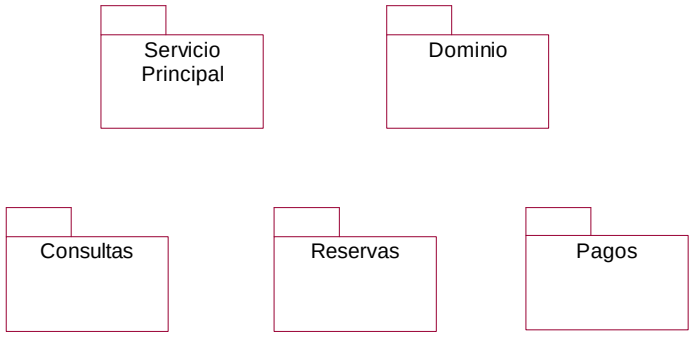

Figura 7.61 Módulos adicionales del módulo *Servicios*.

ServicioPrincipal

El módulo *ServicioPrincipal* está compuesto por las clases:

?? **InterfaceBaseDatosReserva** - Clase Borde. La información del sistema de reservaciones de vuelo se almacena en la base de datos de reservas la cual se accesa mediante la borde de la base de datos de reservas. Esto permite generar consultas, reservas y pago de reservas de manera dinámica.

?? **PantallaServicio** - Clase Borde. Pantalla de servicios (*P*-2).

?? **ManejadorServicio** - Clase Control. El manejador de servicios se encarga de enviar las peticiones particulares de servicios a los manejadores espacializados para consulta, reserva y compra.

Dominio

El módulo *Dominio* está compuesto por las clases:

?? **Vuelo** - Clase Entidad. Se denomina por medio de un número. El vuelo tiene como origen un aeropuerto en una ciudad y tiene como destino un aeropuerto de otra ciudad. Un vuelo puede tener múltiples escalas y múltiples vuelos se relacionan por medio de conexiones. El vuelo pertenece a una aerolínea y puede operar varios días a la semana teniendo un horario de salida y otro de llegada.

?? **Reservación** - Clase Entidad. Para poder tomar un vuelo es necesario contar con una reservación previa, la cual debe pagarse antes de una fecha límite, que puede ser el propio día del vuelo. Una reservación puede hacerse para múltiples vuelos y múltiples pasajeros. La reservación cuenta con una clave identificando un récord de reservación particular.

?? **Horario** - Clase Entidad. El horario de un vuelo se determina por su hora de salida y hora de llegada durante los días que opera.

?? **Aerolínea** - Clase Entidad. La aerolínea provee servicio de múltiples vuelos entre diferentes ciudades bajo diferentes horarios. La aerolínea se identifica por un nombre.

?? **Aeropuerto** - Clase Entidad. El aeropuerto sirve como origen, destino y escalas de un vuelo. El aeropuerto se encuentra en una ciudad de un país determinado.

?? **Tarifa** - Clase Entidad. Los diferentes vuelos tienen múltiples tarifas para compra de boleto, variando según la clase de boleto, si son de ida o de ida y vuelta, y dependiendo de las diversas restricciones y ofertas existentes.

?? **Asiento** - Clase Entidad. Una reservación de vuelo puede incluir la asignación de asiento, especificada mediante una fila y un número. El número de asientos disponibles en un vuelo particular dependen del tipo de avión que opere ese día.

?? **Pasajero** - Clase Entidad. Para poder hacer una reservación se requiere dar el nombre del pasajero. Varios pasajeros pueden aparecer bajo una sola reservación.

?? **Avión** - Clase Entidad. Un vuelo en una fecha determinada se hace en un tipo de avión particular. El tipo de avión define la cantidad máxima de pasajeros que pueden viajar en ese vuelo para esa fecha.

?? **ViajeroFrecuente** - Clase Entidad. El pasajero tiene la opción de acumular millas para un vuelo particular si cuenta con una tarjeta de viajero frecuente para la aerolínea correspondiente.

Consultas

El módulo *Consultas* se divide en los siguientes módulos: *ConsultasPrincipal*, *Horarios*, *Tarifas* y *Estado*, como se muestra en la Figura 7.62.

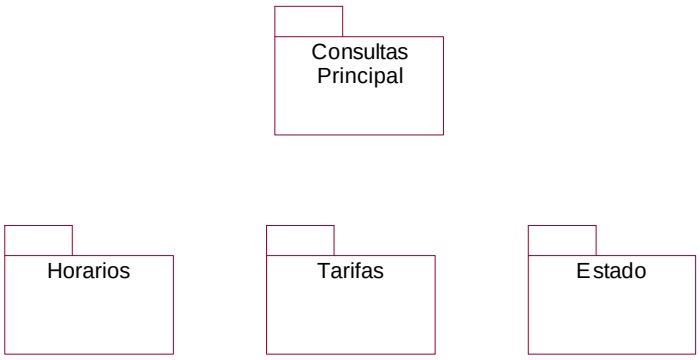

Figura 7.62 Módulos adicionales del módulo *Consultas*.

ConsultasPrincipal

El módulo *ConsutlasPrincipal* está compuesto por las clases:

?? **PantallaConsultas** - Clase Borde. Pantalla de presentación de consultas (*P*-7).
?? **ManejadorConsultas** - Clase Control. El manejador de consulta se encarga de enviar las peticiones de consulta particular a los manejadores de consulta especializados.

Horarios

El módulo *Horarios* está compuesto por las clases:

?? **PantallaConsultaHorarios** - Clase Borde. Pantalla de presentación de consulta de horarios (*P*-8).
?? **PantallaResultadoHorarios** - Clase Borde. Pantalla de devolución de consulta de horarios (*P*-9).
?? **ManejadorConsultaHorarios** - Clase Control. El manejador de consulta de horarios se encarga de controlar las peticiones de consulta de horarios.

Tarifas

El módulo *Tarifas* está compuesto por las clases:

?? **PantallaConsultaTarifas** - Clase Borde. Pantalla de presentación de consulta de tarifas (*P*-10).
?? **PantallaResultadoTarifas** - Clase Borde. Pantalla de devolución de consulta de tarifas (*P*-11).
?? **ManejadorConsultaTarifas** - Clase Control. El manejador de consulta de tarifas se encarga de controlar las peticiones de consulta de tarifas.

Estado

El módulo *Estado* está compuesto por las clases:

?? **PantallaConsultaEstado** - Clase Borde. Pantalla de presentación de consulta de estado (*P*-12).
?? **PantallaResultadoEstado** - Clase Borde. Pantalla de devolución de consulta de estado (*P*-13).
?? **ManejadorConsultaEstado** - Clase Control. El manejador de consulta de estado se encarga de controlar las peticiones de consulta de estado.

Reservas

El módulo *Reservas* está compuesto por las clases:

?? **PantallaClaveReservas** - Clase Borde. Pantalla de solicitud de clave de reservas (*P*-14).
?? **PantallaCrearReservaVuelos** - Clase Borde. Pantalla de solicitud de reservas (*P*-15).
?? **PantallaRecordReservaVuelos** - Clase Borde. Pantalla de devolución de reservas (*P*-16).
?? **ManejadorReservas** - Clase Control. El manejador de reserva se encarga de enviar las solicitudes de reserva a la base de datos del sistema de reservaciones.

Pagos

El módulo *Pagos* está compuesto por las clases:

?? **PantallaPagarRegTarjeta** - Clase Borde. Pantalla de solicitud de pago de reservas (*P*-17).
?? **PantallaReembolsarRegTarjeta** - Clase Borde. Pantalla de solicitud de reembolso de pago (*P*-18).

?? **ManejadorPagos** - Clase Control. El manejador de compra se encarga de enviar las solicitudes de compra de boleto a la base de datos del sistema de reservaciones.