Programación 4

CASO DE ESTUDIO :: ANÁLISIS

OBJETIVO

En este documento se presenta un caso de estudio que servirá como guía práctica para la introducción de la etapa de análisis, tanto para las actividades de Modelado de Dominio como de Especificación del Comportamiento del Sistema.

DESCRIPCIÓN DE LA REALIDAD

Se desea construir una aplicación web que permita la reserva y venta de pasajes de avión. La aplicación debe manejar información de las personas que realizan las reservas y compran los pasajes. De cada persona se sabe su nombre (que lo identifica), teléfono y edad. Las mismas pueden reservar o comprar múltiples pasajes, pero para eso deben ser mayores de 18 años. Las personas deben primero reservar pasajes y en ese caso se registra la fecha de reserva y el costo de la misma en dicho momento.

Cuando la persona compra la reserva queda responsable de realizar el pago, sea en efectivo o con tarjeta de crédito. En caso de pagar en efectivo, se aplica un descuento porcentual que es el mismo para todos los pagos en efectivo. En caso contrario es necesario registrar el número de la tarjeta y la cantidad de pagos que se realizarán.

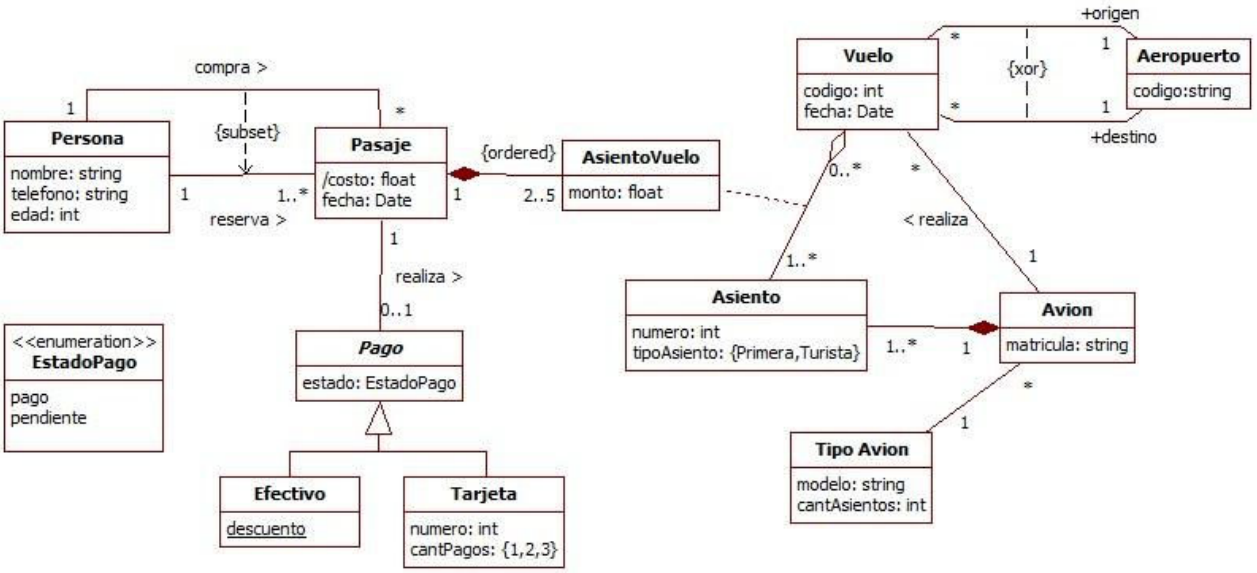
Existen diferentes modelos de avión, cada modelo con una cantidad de asientos. De cada avión se registra una matrícula que permite identificarlos. Cada asiento está identificado por un número y se conoce además su ubicación, sea en primera clase o en clase turista.

De cada vuelo (tramo) identificado con un código, se conoce su fecha de realización, el aeropuerto de origen y el de destino (identificados por un código de tres letras), no pudiendo ser ambos el mismo aeropuerto. A cada vuelo se le asigna un avión específico pudiendo reservarse/venderse los asientos disponibles de dicho avión. El costo de realizar un vuelo depende además del asiento en el que se vaya, por ejemplo, los asientos de primera valen más que los de turista, e incluso dentro de primera pueden haber diferencias según la ubicación del asiento.

Los pasajes que una persona reserva/compra serán una secuencia ordenada de entre 2 y 5 trayectos a los cuales se les asignará un asiento concreto. El costo del pasaje será la suma del costo de los asientos que se reserven. Como forma de mejorar el trato con los clientes, si un viaje tiene más de 3 vuelos, se le aplica el descuento de pago en efectivo aún si paga con tarjeta.

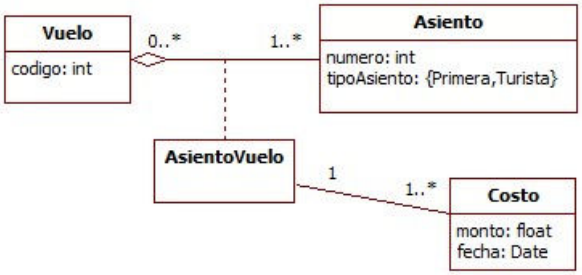
MODELADO DE DOMINIO

Un posible modelo para la realidad planteada es el siguiente.



Algunos comentarios sobre el modelo:

- La asociación entre Pasaje y AsientoVuelo representa la ocupación del Vuelo, es decir, si existe un link entre dos instancias de estas clases indica que el asiento correspondiente está ocupado para ese vuelo.
- En tanto un pasaje no se compre no habrá un link entre Pasaje y Pago
- En caso de querer manejar un costo de asientos variable a lo largo del tiempo y no perder los costos con los cuales se compró un pasaje, es necesario realizar la siguiente modificación en el modelo. La nueva clase Costo es la que debería asociarse con Pasaje en lugar de la clase AsientoVuelo.



Algunas de las restricciones presentes en el modelo son las siguientes:

- El nombre de una persona la identifica (unicidad de atributos)
- La edad de una persona debe ser mayor o igual a 18 (dominio de atributos)
- Los asientos asignados al vuelo que realiza un avión son asientos de ese avión (integridad circular)
- El costo de un pasaje es la suma de los montos de las instancias de AsientoVuelo asociadas menos el descuento correspondiente (atributos calculados)
- Si un pasaje tiene más de tres instancias de AsientoVuelo asociadas y el pasaje fue comprado y pago con tarjeta, entonces al costo se le aplica el descuento de pago en efectivo (regla de negocio)

ESPECIFICACIÓN DEL COMPORTAMIENTO DEL SISTEMA

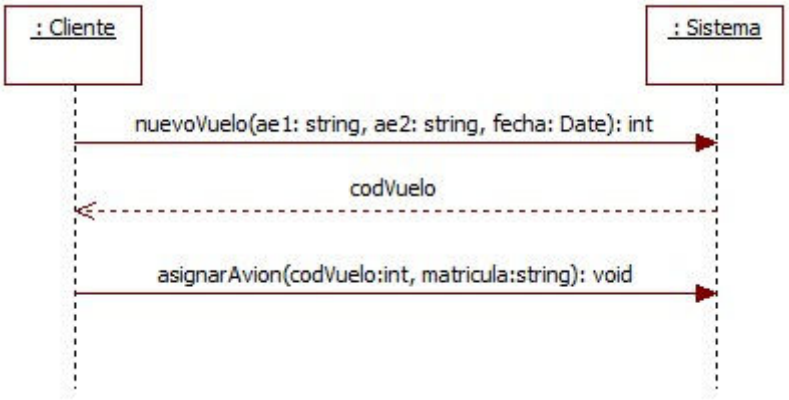
A continuación se presentan dos casos de uso del sistema junto con sus respectivos Diagramas de Secuencia del Sistema y algunos de los contratos de las operaciones involucradas. En los casos de uso se puede apreciar a dos actores diferentes: el Cliente que interactúa con el sistema para la reserva y compra de pasajes, y el Administrador encargado de las tareas de mantenimiento.

Nota: Los Diagramas de Secuencia del Sistema (DSS) son una extensión de los Diagramas de Secuencia de UML en donde aparecen los actores enviando mensajes al sistema. Dado que no son parte del estándar, muchas de las herramientas de modelado no permiten modelar a los actores, salvo mediante un objeto que lo represente. No obstante, para nuestros propósitos esto será suficiente.

Alta de Vuelo

Nombre	Alta de Vuelo
Actores	Administrador
Descripción	El caso de uso comienza con el Administrador que solicita al Sistema la creación de un nuevo vuelo indicando los aeropuertos de origen y destino del mismo, así como la fecha de realización del vuelo. El Sistema da de alta al vuelo y retorna un código autogenerado que identifica al mismo. A continuación, el Administrador solicita al Sistema que asigne un avión al vuelo recientemente creado indicándole la matrícula del avión. El Sistema relaciona el vuelo con el avión y con ello define los asientos que estarán disponibles para reservar.

El DSS correspondiente al caso de uso anterior se muestra a continuación. Notar que en él mismo, el Sistema **NO TIENE MEMORIA**. Esto se ve reflejado por la necesidad de indicar en el segundo mensaje el código del vuelo dado de alta a consecuencia del primer mensaje.



Los contratos de las operaciones correspondientes a los mensajes del DSS anterior son los siguientes.

Operación	`nuevoVuelo(ae1:string, ae2:string, fecha:Date):int`
Descripción	Da de alta un nuevo vuelo
Pre y postcondiciones	

`Pre`: Existe una instancia de `Aeropuerto` con atributo `codigo = ae1`
`Pre`: Existe una instancia de `Aeropuerto` con atributo `codigo = ae2`
`Post`: Se creó una instancia de `Vuelo` con atributo `fecha = fecha` y atributo `codigo` autogenerado por el sistema
`Post`: Se creó un link entre la instancia de `Vuelo` y la instancia de `Aeropuerto` con atributo `codigo = ae1` con rol `origen`
`Post`: Se creó un link entre la instancia de `Vuelo` y la instancia de `Aeropuerto` con atributo `codigo = ae2` con rol `destino`
`Post`: Se retornó el atributo `codigo` de la instancia de `Vuelo` creada	

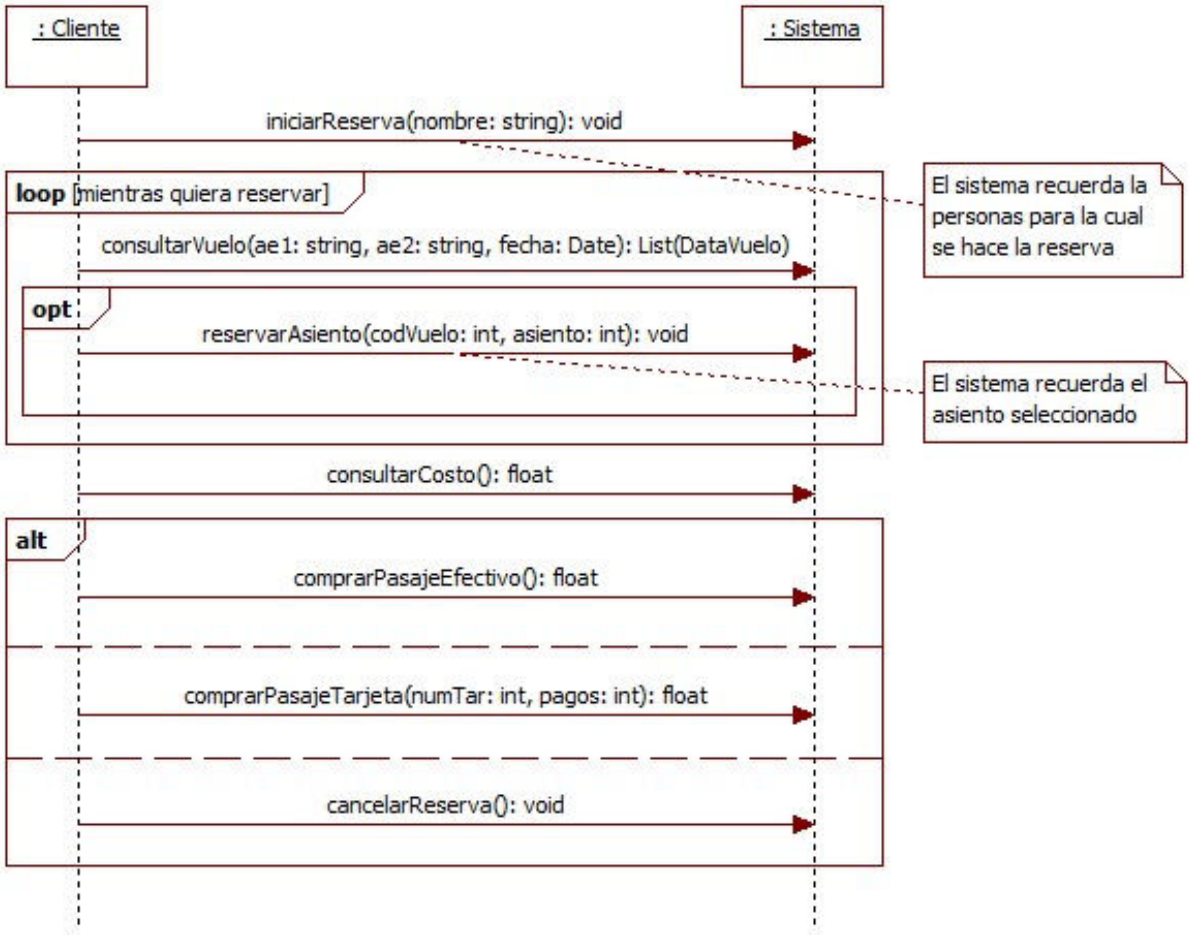
Operación	`asignarAvion(codVuelo:int, matricula:string)`
Descripción	Asigna un avión a un vuelo
Pre y postcondiciones	

`Pre`: Existe una instancia de `Vuelo` con atributo `codigo = codVuelo`
`Pre`: Existe una instancia de `Avion` con atributo `matricula = matricula`
`Pre:` No existe un link entre la instancia de `Vuelo` y la de `Avion`
`Post`: Se creó un link entre la instancia de `Vuelo` y la de `Avion`
`Post`: Para cada instancia de `Asiento` asociada a la instancia de `Avion`, se creó una instancia de la clase de asociación `AsientoVuelo` entre el `Asiento` y el `Vuelo` con atributo `monto = 0`	

Reserva y Compra de Pasaje

Nombre	Reserva y Compra de Pasaje
Actores	Cliente
Descripción	El caso de uso comienza con el Cliente que solicita al Sistema el inicio de una reserva indicándole su nombre (que ya está registrado en el Sistema). Luego el Cliente comienza un ciclo de consulta de vuelos y reserva de los mismos. En primer lugar consulta con el Sistema los vuelos definidos entre dos aeropuertos a partir de una fecha dada y el Sistema le retorna la lista de códigos de vuelo junto con la lista de asientos disponibles para cada uno de ellos. En segundo lugar, opcionalmente, el Cliente reserva un asiento para un vuelo determinado y el Sistema registra la reserva. Al finalizar este ciclo el Cliente solicita al Sistema el costo final de la reserva realizada. Finalmente, el Cliente le indica al Sistema si comprará el pasaje, ingresando en este caso si pagará con tarjeta o en efectivo, o si cancelará la reserva. En el primer caso el Sistema retorna el costo final a pagar por parte del Cliente.

El DSS correspondiente al caso de uso anterior se muestra a continuación. Notar que en él mismo, el Sistema **TIENE MEMORIA**. Esto se ve reflejado en los comentarios.



Los contratos de las operaciones correspondientes a los mensajes del DSS anterior son los siguientes.

Operación	`iniciarReserva(nombre:string)`
Descripción	Se inicia una reserva
Pre y postcondiciones	

`Pre`: Existe una `Persona` con atributo `nombre = nombre`
`Post`: Se crea una instancia de `Pasaje` con atributo `fecha = hoy` y se crea un link de la asociación `reserva` entre la instancia de `Persona` y la de `Pasaje`
`Post`: El sistema recuerda la instancia de `Persona` y la de `Pasaje`	

Operación	consultarVuelo(ae1:string, ae2:string, fecha:Date): List(DataVuelo)
Descripción	Se consulta la disponibilidad de vuelos
Pre y postcondiciones	
Pre: Existe una instancia de Aeropuerto con atributo código = ae1 Pre: Existe una instancia de Aeropuerto con atributo código = ae2 Post: Se devuelve una lista de tipo DataVuelo donde cada una representa información sobre las instancias de Vuelo con atributo fecha >= fecha, con links a las instancias de Aeropuerto anteriores con rol origen y destino, respectivamente. Cada DataVuelo contiene el valor del atributo codigo de la instancia de Vuelo y la lista de valores de los atributos numero de las instancias de Asiento asociadas y disponibles.	

Operación	reservarAsiento(codVuelo:int, asiento:int)
Descripción	Se reserva un asiento para un vuelo determinado
Pre y postcondiciones	
Pre: Existe una instancia de Pasaje recordada por el sistema Pre: Existe una instancia de Vuelo con atributo código = codVuelo Pre: Existe una instancia de Asiento con atributo numero = asiento asociada a la instancia de Vuelo Pre: No existe un link entre la instancia de AsientoVuelo, asociada a las instancias de Vuelo y Asiento, y una instancia de Pasaje. (el asiento está disponible) Pre: El sistema recuerda menos de cinco instancias de AsientoVuelo Post: Se crea un link entre la instancia de Pasaje recordada por el sistema y la instancia de AsientoVuelo	

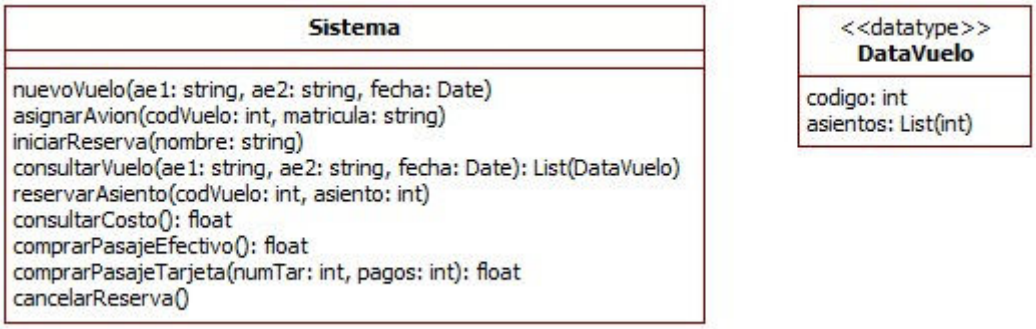
Operación	consultarCosto():float
Descripción	Se consulta el costo final del pasaje reservado sin los descuentos
Pre y postcondiciones	
Pre: Existe una instancia de Pasaje recordada por el sistema Post: Se devuelve la suma de los valores del atributo monto de las instancias de AsientoVuelo asociadas a la instancia de Pasaje	

Operación	comprarPasajeEfectivo():float
Descripción	Se ingresa la responsabilidad de pago en efectivo
Pre y postcondiciones	
Pre: El sistema recuerda una instancia de Persona y una de Pasaje asociada a entre dos y cinco instancias de AsientoVuelo Pre: No existe un link de la asociación compra entre la instancia de Persona recordada por el sistema y la instancia de Pasaje Pre: No existe un link entre la instancia de Pasaje y una instancia de Pago	

Post: Se crea un link de la asociación compra entre la instancia de Persona recordada por el sistema y la instancia de Pasaje Post: Se crea una instancia de Efectivo con atributo estado = pendiente y se crea un link entre dicha instancia y la instancia de Pasaje Post: Se devuelve la suma de los valores del atributo monto de las instancias de AsientoVuelo asociadas a la instancia de Pasaje con el descuento correspondiente al atributo descuento de Efectivo	
Operación	comprarPasajeTarjeta(numTar:int, pagos:int):float
Descripción	Se ingresa la responsabilidad de pago con tarjeta
Pre y postcondiciones	
Pre: El sistema recuerda una instancia de Persona y una de Pasaje asociada a entre dos y cinco instancias de AsientoVuelo Pre: No existe un link de la asociación compra entre la instancia de Persona recordada por el sistema y la instancia de Pasaje Pre: No existe un link entre la instancia de Pasaje y una instancia de Pago Post: Se crea un link de la asociación compra entre la instancia de Persona recordada por el sistema y la instancia de Pasaje Post: Se crea una instancia de Tarjeta con atributo estado = pendiente, atributo numero = numTar y atributo cantPagos = pagos y se crea un link entre dicha instancia y la instancia de Pasaje Post: Se devuelve la suma de los valores del atributo monto de las instancias de AsientoVuelo asociadas a la instancia de Pasaje	

Operación	cancelarReserva()
Descripción	Se cancela la reserva realizada
Pre y postcondiciones	
Pre: El sistema recuerda una instancia de Persona y una de Pasaje asociada a entre dos y cinco instancias de AsientoVuelo Post: Se elimina la instancia de Pasaje y los links entre ésta y las instancias de Persona y AsientoVuelo recordadas por el sistema Post: El sistema deja de recordar información	

Finalmente se muestra a continuación la definición del datatype DataVuelo utilizado en el DSS anterior, así como un resumen de las operaciones que fueron definidas para el sistema en su conjunto. Estas operaciones son las que deberán ser diseñadas.


Sistema
nuevoVuelo(ae1: string, ae2: string, fecha: Date)
asignarAvion(codVuelo: int, matricula: string)
iniciarReserva(nombre: string)
consultarVuelo(ae1: string, ae2: string, fecha: Date): List(DataVuelo)
reservarAsiento(codVuelo: int, asiento: int)
consultarCosto(): float
comprarPasajeEfectivo(): float
comprarPasajeTarjeta(numTar: int, pagos: int): float
cancelarReserva()
<<datatype>>
DataVuelo
codigo: int
asientos: List(int)