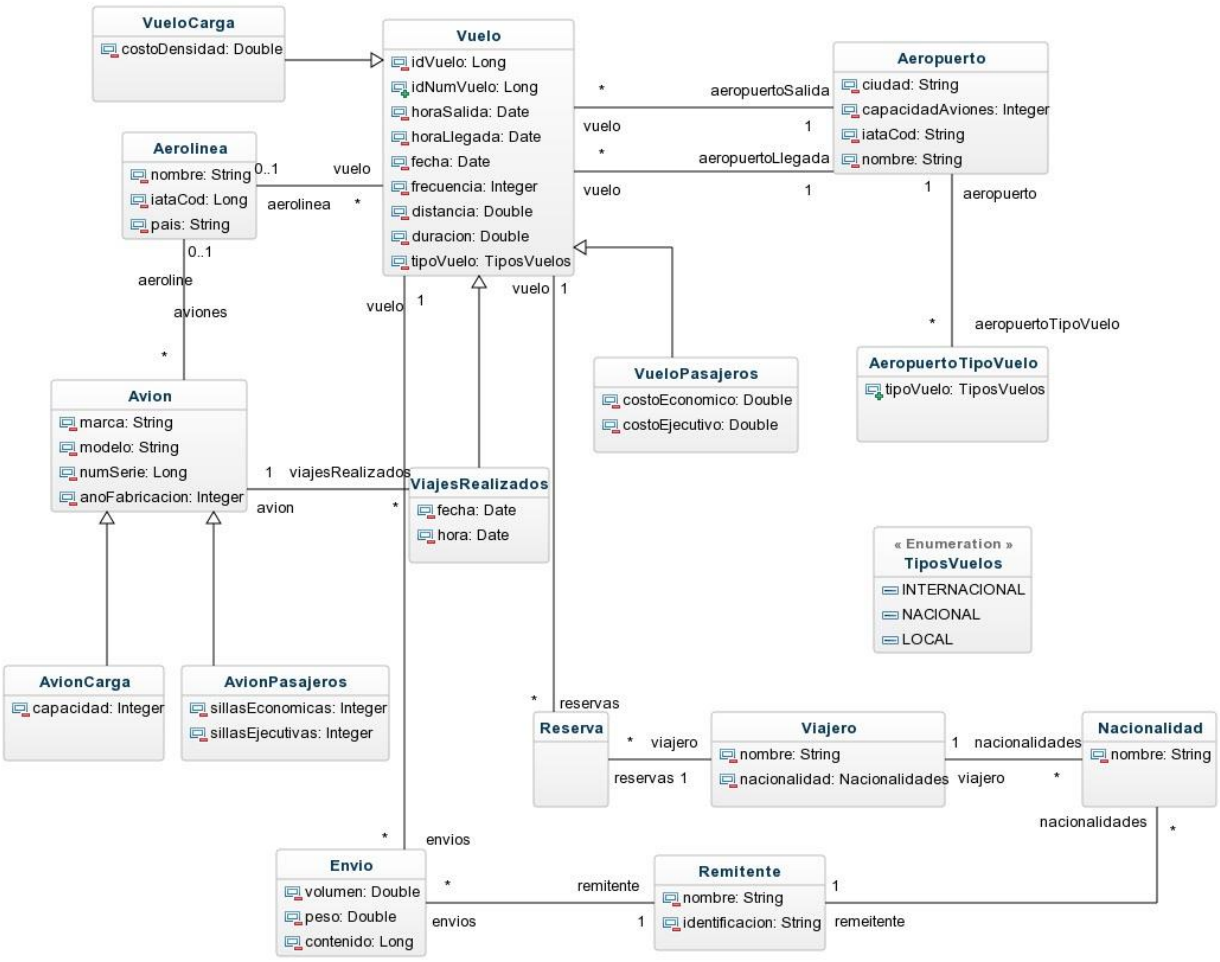
Iteración 3 Sistemas transaccionales

Rogelio García 201326488, Stephannie Jiménez 201423727

1.

Frente al modelo UML anterior, la clase “Viajes realizados” ahora hereda de Vuelos y se mejoró la organización para mayor claridad.



2. Para la inclusión del concepto de escalas, se debió modificar la tabla de reservas. En este nuevo diseño, se modificó la llave primaria de la tabla “Reservas” para que incluyera el id de la reserva idReserva y el id del vuelo idVuelo. De esta manera, un solo idReserva puede estar presente en varios vuelos, dando a entender que hacen parte de un mismo trayecto dividido por escalas. Si se quiere

revisar las reservas asociadas a un vuelo, se busca dicho idVuelo, y si, al contrario, se quiere eliminar la reserve de un usuario que está presente en más de un vuelo, se busca por idReserva.

Por otro lado, los nuevos requerimientos y su lógica se explican a continuación:

- RF12: Registrar reserva de un viajero
  En este caso, se ingresa el id del viajero, el origen, destino, la aerolínea y el número de sillas. Luego se procede a buscar un vuelo directo o varios vuelos (escalas). Al encontrarlos, se realizan las respectivas reservas. Para realizar las reservas, se utiliza el id del viajero, el número de sillas y los vuelos que se encontraron anteriormente.

  Nivel de aislamiento: SERIALIZABLE.

- RF13: Cancelar reserva de un viajero en un vuelo
  Al cancelar la reserva de un viajero en un vuelo, se especifica el id de la reserva y el id del vuelo. Se procede a eliminar las respectivas entradas.

  Nivel de aislamiento: SERIALIZABLE.

- RF14: Cancelar reserva de un viajero
  Al cancelar la reserva de un viajero, se especifica únicamente el id de la reserva y esta se elimina de todas las entradas. Es decir, si, por ejemplo, hay un viaje con 2 escalas, existen 3 reservas que tienen el mismo idReserva y diferente idVuelo. Todas estas tuplas serían eliminadas.

  Nivel de aislamiento: SERIALIZABLE.

- RF15: Cancelar viaje
  Cancelar un vuelo consiste en:
  1. Recibir el id del vuelo.
  2. Guardar el origen, el destino y la aerolínea del vuelo.
  3. Buscar todas las reservas de ese vuelo.
  4. Organizar las reservas por número de escalas que tengan y guardar las sillas que habían solicitado los viajeros.
  5. Eliminar el viaje.
  6. Realizar nuevamente las reservas con la información de cada viajero, el número de sillas solicitadas y la aerolínea, dado que haya capacidad suficiente.

  Nivel de aislamiento: SERIALIZABLE.

- RF16: Registrar reserva de un vuelo para grupos de viajeros
  Para registrar la reserva de un vuelo para varios viajeros:
  1. El usuario viajero ingresa la información de varios viajeros junto con la información sobre la carga que transportan.
  2. El sistema procede a registrar a los viajeros y los remitentes.
  3. Se procede a revisar la disponibilidad del viaje de carga y del viaje de pasajeros.
  4. Se realiza la reserva en el mismo vuelo para cada pasajero, y se realiza el envío de la carga al mismo destino.

  Nivel de aislamiento: SERIALIZABLE.

- RFC5: Consultar viajes
  Para consultar los viajes se tienen dos variantes:
  - Usuario viajero: Un viajero solo puede consultar la información de sus propios viajes. En este se encuentra información por cada viaje que haya realizado: origen, destino, idVuelo, nombreViajero, idViajero, codigoAerolinea, fecha de salida, código y nombre del aeropuerto de salida, fecha de llegada, código y nombre del aeropuerto de llegada, y por último sillas reservadas y tiempo de viaje.
  - Usuario gerente: Recibe la misma información que el viajero, pero para todos los viajeros registrados en el sistema.

  Nivel de aislamiento: READ-ONLY

- RFC6: Consultar aeronaves
  Para consultar aeronaves, se tienen dos variantes:
  - Usuario aerolínea: Por cada viaje de cada aeronave de la aerolínea recibe: Marca, modelo, número de serie, año de fabricación, nombre de la aerolínea y tiempo de vuelo.
  - Usuario gerente: Recibe la misma información pero para todos los vuelos y aeronaves de todas las aerolíneas.

  Nivel de aislamiento: READ-ONLY

  BONO:

  1. El manejo transaccional que puede realizar el programador de la aplicación consiste en:
     a. Manejar el nivel de aislamiento que tiene cada transacción con comandos como SET TRANSACTION READ ONLY, SET TRANSACTION ISOLATION LEVEL SERIALIZABLE, SET TRANSACCION ISOLATION LEVEL READ COMMITED, SELECT … FOR UPDATE.
     b. Manejo de savepoints, rollbacks y commits. De esta manera el programador sabe qué acciones revertir en caso de encontrar alguna falla o inconsistencia.

  Se podría decir que el manejo transaccional que tiene el programador de la aplicación es una versión más simple del manejo que tiene un servidor de aplicaciones. Este último se encarga de elementos de transaccionalidad de manera transparente al programador. Por ejemplo, para que una tabla que contenga llaves foráneas borre las tuplas determinadas cuando se borre una llave de la tabla origen, tendría que modificar las restricciones de las tablas directamente en el ambiente de la base de datos (ON DELETE CASCADE). Si se tiene un contenedor/servidor de aplicaciones, esta situación se manejaría mediante anotaciones (CascadeType.All). Mediante anotaciones también se definen las relaciones entre tablas y demás.

  El manejo transaccional por parte del programador puede ser más flexible y detallado, mientras que por parte del contenedor sería más sencillo.

  2. La interfaz más bonita es la nuestra. Prueba:


THE GOLDEN


THE GOLDEN