# EJEMPLO 2 (página 265 libro)

**Ejemplo 2:** Se desea realizar el análisis de un sistema de gestión informática de una pequeña agencia de viajes que oferta viajes a sus clientes. El sistema debe proporcionar una ventana inicial con una serie de menús que abrirán paso al resto de ventanas de la aplicación que permitirán realizar las siguientes acciones:

- La gestión de las reservas de viajes para realizar reservas, modificar reservas, consultar reservas, borrar reservas y generar e imprimir facturas.
- El mantenimiento de datos de clientes. Para mantener actualizados los datos se realizarán operaciones de consulta, altas, bajas y modificaciones de datos de clientes, y además debe permitir generar listados de clientes.
- Mantenimiento de datos de viajes. Para mantener actualizados los datos se realizarán operaciones de consulta, altas, bajas, modificaciones e informes de viajes.

Disponemos de una base de datos donde están almacenados los datos de los clientes, los viajes, las reservas, las fechas de viaje, los datos son los siguientes:

- Datos de clientes son: código-cliente, nombre, tlf y dirección.
- Datos de viajes son: código, nombre, plazas y precio.
- Datos de las reservas son: número de reserva y estado de la reserva.
- Un cliente puede realizar muchas reservas, y una reserva es de un cliente.
- Igualmente, de un viaje se pueden realizar muchas reservas, y una reserva pertenecerá a un viaje.
- Los viajes se ofertan en varias fechas de viaje, de estas fechas se necesita saber la fecha de comienzo y la fecha de fin. Estas fechas pueden ser compartidas por varios viajes.
- También se cuenta con la información de un catálogo de viajes. Datos del catálogo son código, destino, procedencia, temporada, precio. Los viajes se crean a partir del catálogo.

En primer lugar, vamos a diseñar este diagrama de clase en base a *paquetes*.

- **Paquete *Interfaz*:**
  Lo constituyen las diferentes ventanas de la aplicación.
- **Paquete *Control*:**
  Lo forma la lógica en sí de la aplicación, es decir, los tipos de operaciones con los datos del os que se disponen.
- **Paquete *Entidad*:**
  Contendrá las diferentes clases asociadas a la BBDD.

1. **Clases de tipo Entidad.**
   Creamos las cinco clases llamadas:
   - Catálogo
   - Reserva
   - Viaje
   - Fecha_Viaje
   - Cliente

   Dentro de cada uno, en la pestaña de propiedades, vamos pulsando el botón de *Nuevo Atributo* tantas veces como atributos le correspondan a cada clase, con visibilidad *privada* (signo – a la izquierda del atributo), detallando el tipo de variable que sería en un lenguaje de programación.

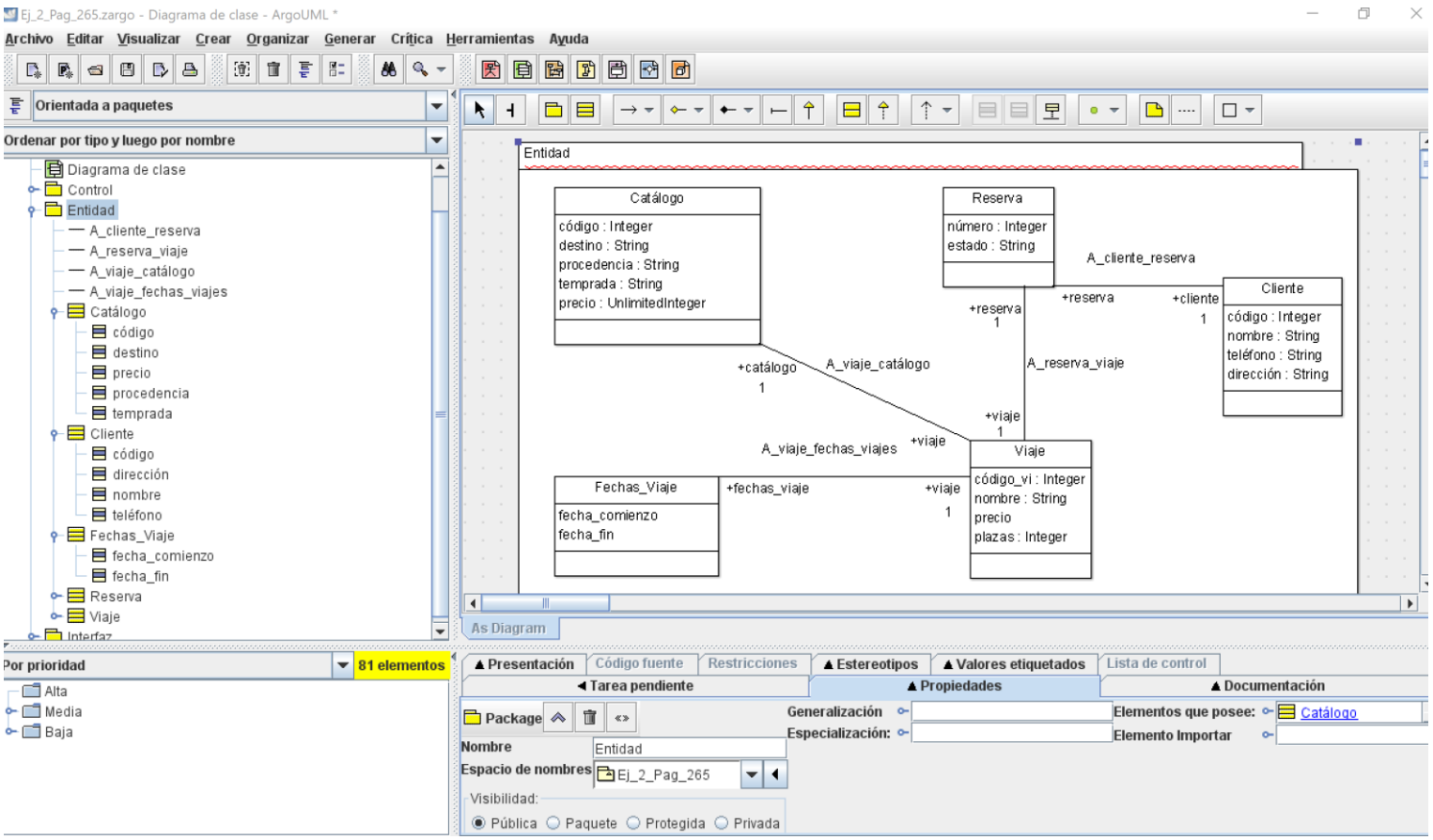
   - **Catálogo**
     - Código
     - Destino
     - Procedencia
     - Temprada
     - Precio
   - **Reserva**
     - Número
     - Estado
   - **Viaje**
     - Código_vi
     - Nombre
     - Precio
     - Plazas
   - **Fecha_Viaje**
     - Fecha_comienzo
     - Fecha_fin
   - **Cliente**
     - Código
     - Nombre
     - Teléfono

   - **Catálogo**
     - Código: Interger
     - Destino: String
     - Procedencia: String
     - Temprada: String
     - Precio: Float
   - **Reserva**
     - Número: Interger
     - Estado: String
   - **Viaje**
     - Código_vi: Interger
     - Nombre: String
     - Precio: Float
     - Plazas: Interger
   - **Fecha_Viaje**
     - Fecha_comienzo: Date
     - Fecha_fin: Date
   - **Cliente**
     - Código: Interger
     - Nombre: String
     - Teléfono: String

Ahora debemos crear la **Asociación y Multiplicidad** entre las clases que hemos creado

- La **navegabilidad** es bidireccional en todos los casos, es decir, cualquier clase conoce la existencia de las de las demás.
- Las **multiplicidades** son:
  - Entre **Catálogo y Viaje:**
    - Un catálogo tiene varios viajes (*)
    - Un viaje es tenido por un catálogo (1)
    - Multiplicidad total (1..*)
  - Entre **Reserva y Cliente:**
    - Varias reservas son hechas por un cliente (1)
    - Un cliente hace varias reservas (*)
    - Multiplicidad total (1..*)
  - Entre **Reserva y Viaje:**
    - Varias reservas forman un viaje (1)
    - Un viaje está formado por varias reservas (*)
    - Multiplicidad total (1..*)
  - Entre **Viaje y Fecha_Viaje:**
    - Un viaje tiene varias fechas (*)
    - Una fecha tiene varios viajes (*)
    - Multiplicidad total (M..N)

El resultado queda reflejado en la siguiente imagen:

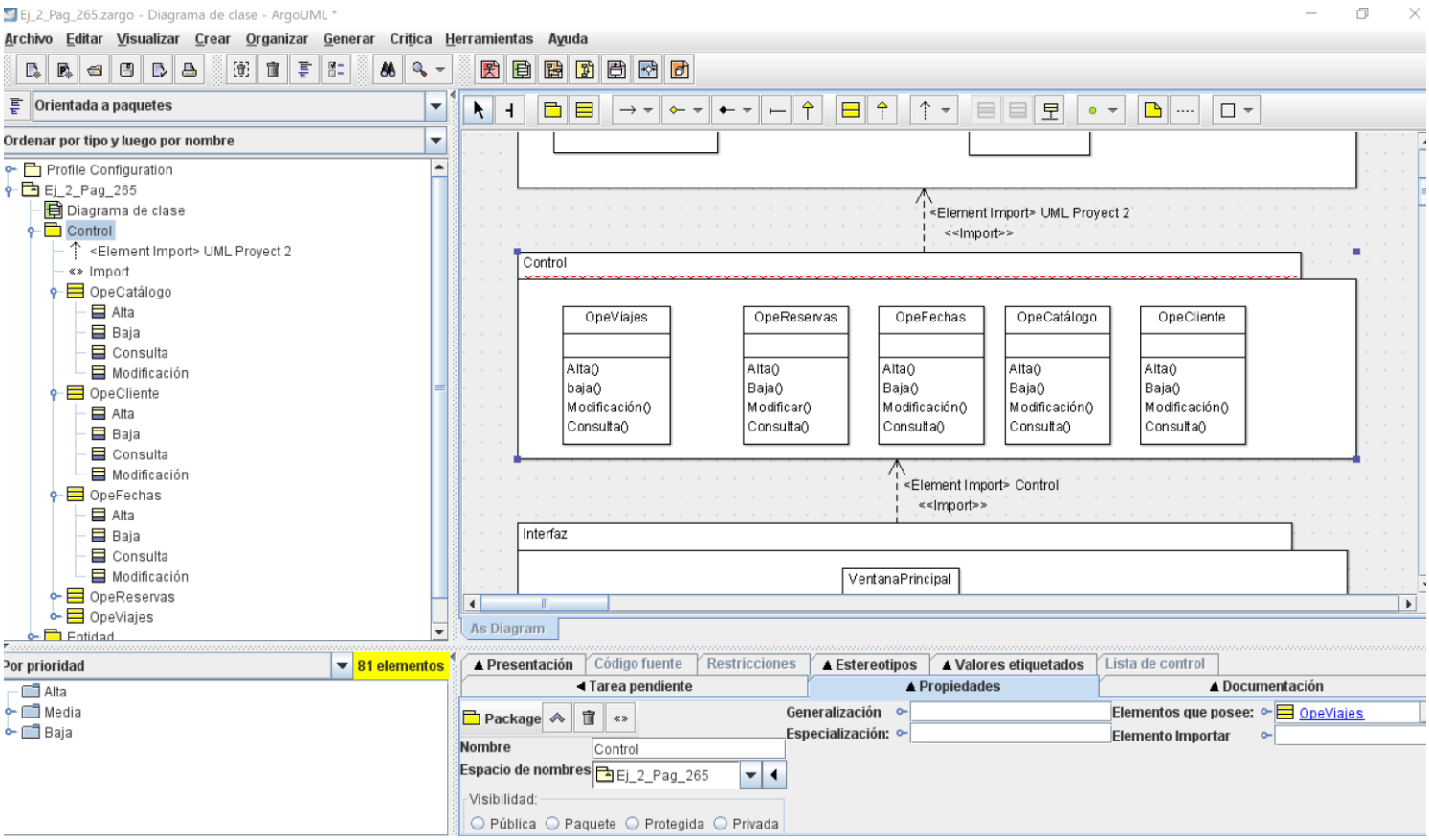


2. Ahora **creamos el siguiente paquete**, el de las clases **de tipo Control**, en el que sus clases, no tendrán atributos, si no que contendrán métodos, los cuales al contrario que los atributos, éstos serán de visibilidad *pública* (signo + a la izquierda del atributo).

- **OpeViajes**
  - + Alta()
  - + Baja()
  - + Modificación()
  - + Consulta()
- **OpeReservas**
  - + Alta()
  - + Baja()
  - + Modificación()
  - + Consulta()
- **OpeFechas**
  - + Alta()
  - + Baja()
  - + Modificación()
  - + Consulta()
- **OpeCatálogo**
  - + Alta()
  - + Baja()
  - + Modificación()
  - + Consulta()
- **OpeCliente**
  - + Alta()
  - + Baja()
  - + Modificación()
  - + Consulta()

Para terminar con este paquete, debemos unirlo al paquete de Entidad mediante una flecha discontinua sin relleno en el pico, detallando el estereotipo <<Import>>, ya que las diversas acciones que reflejan las clases del paquete Control con sus métodos, necesitan los datos de las clases de Entidad para su cometido, es decir, existe una relación de dependencia entre las clases de ambos paquetes.

El resultado queda reflejado en la siguiente imagen:



3. Por último, sólo faltaría **crear paquete de Interfaz**, el cual contendrá las diferentes ventanas por las cuales el usuario se irá desplazando, el cual estará formado por cuatro clases que son las diferentes ventanas de la aplicación, cada una con una serie de métodos estandarizados y comunes a las demás clases (ventanas) para que el usuario pueda gestionar correctamente sus datos para la reserva de su viaje.
Habrá una superclase que generalizará a tres subclases, cuya **relación de herencia** se plasmará relacionando las tres subclases con la superclase mediante una flecha continua y rellena, que significará que estas tres, se especializan en la superclase.
*Nota: en este paquete también hay que poner en Público todos los métodos (signo +).*

- **VentanaPrincipal**
  - + OpenViajes()
  - + OpenReservas()
  - + OpenClientes()
- **vViajes**
  - + FormInsertar()
  - + FormModificar()
  - + FormBorrar()
  - + FormConsultar()
  - + Informes()
- **vReservas**
  - + FormInsertar()
  - + FormModificar()
  - + FormBorrar()
  - + FormConsultar()
  - + Informes()
- **vClientes**
  - + FormInsertar()
  - + FormModificar()
  - + FormBorrar()
  - + FormConsultar()
  - + Informes()

Para terminar con este paquete, debemos unirlo al paquete de Entidad mediante una flecha discontinua sin relleno en el pico, detallando el estereotipo <<Import>>, ya que las diversas acciones que reflejan las clases del paquete Interfaz con sus métodos, necesitan los datos de las clases de Control para su cometido, es decir, existe una relación de dependencia entre las clases de ambos paquetes