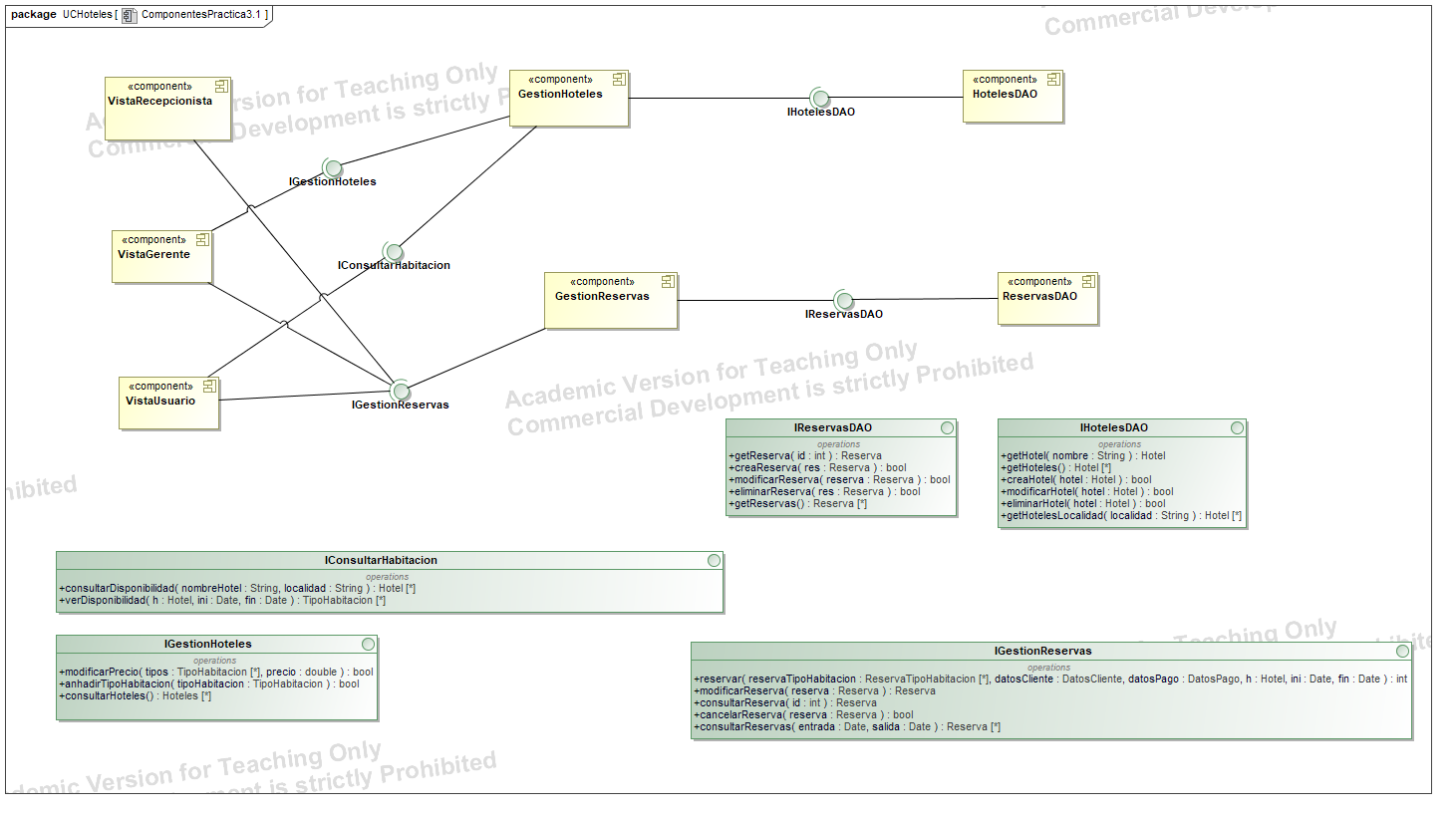
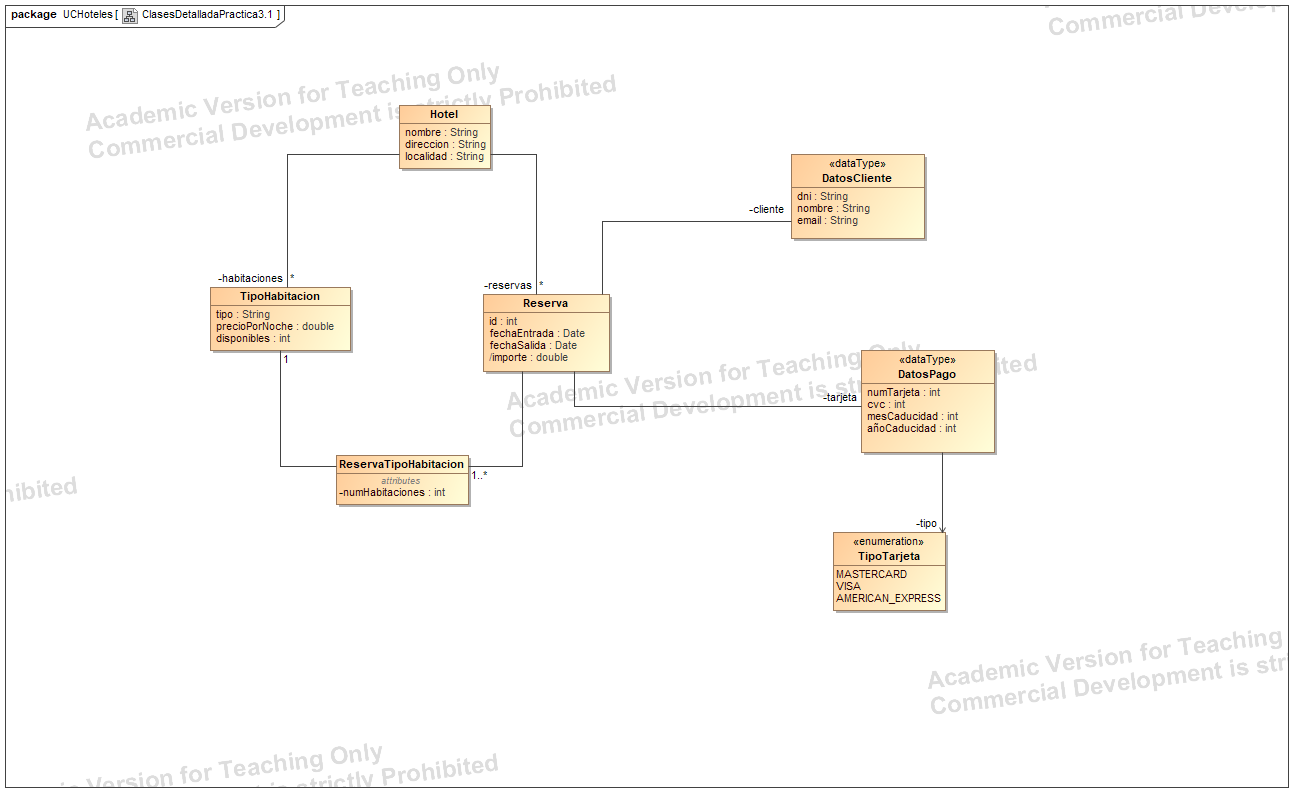
PRACTICA 3.1 -Jesús Saiz Gutiérrez



Como se puede ver en la foto superior, así es como ha quedado la arquitectura del sistema para los casos de uso especificados en UCHoteles.

También se ha retocado el diagrama de clases UML añadiendo los tipos de datos, convirtiendo la relación de ReservaTipoHabitacion en una clase y añadiendo a las relaciones de esta misma las multiplicidades correspondientes.



PLAN DE PRUEBAS

**Pruebas de Aceptación**

Pruebas de aceptación del caso de uso “Reservar”.

##### **Prueba 1: Reserva exitosa con habitaciones disponibles.**

1. El usuario introduce el número de habitaciones deseadas de cada tipo.
2. El usuario selecciona la opción de reservar.
3. El sistema comprueba que existen habitaciones suficientes de cada tipo.
4. El sistema muestra el importe total de la reserva y pide al cliente sus datos.
5. El usuario proporciona sus datos y confirma la reserva.
6. **Se verifica que** el sistema almacena la reserva y muestra el identificador único.

##### **Prueba 2: Reserva con habitaciones no disponibles.**

1. El usuario introduce el número de habitaciones deseadas de cada tipo.
2. El usuario selecciona la opción de reservar.
3. El sistema comprueba que existen habitaciones suficientes de cada tipo.
4. **Se verifica que el sistema** muestra un mensaje informativo sobre la falta de disponibilidad.

##### **Prueba 3: Reserva con problemas de conexión a la base de datos al consultar disponibilidad.**

1. El usuario introduce el número de habitaciones deseadas de cada tipo.
2. El usuario selecciona la opción de reservar.
3. El sistema intenta comprobar que existen habitaciones suficientes de cada tipo.
4. Se verifica que el sistema muestra un mensaje informativo sobre la incapacidad de conectar con la base de datos.

##### **Prueba 4: Reserva con problemas de conexión a la base de datos al almacenar la reserva.**

1. El usuario introduce el número de habitaciones deseadas de cada tipo.
2. El usuario selecciona la opción de reservar.
3. El sistema comprueba que existen habitaciones suficientes de cada tipo.
4. El sistema muestra el importe total de la reserva y pide al cliente sus datos.
5. El usuario proporciona sus datos y confirma la reserva.
6. **Se verifica que** el sistema muestra un mensaje informativo ya que no se ha podido almacenar la reserva en la base de datos.

**Pruebas Unitarias y de Integración :**

**Los casos de prueba unitarios y de integración son los mismos, lo único que cambia es la forma en la que se realizan las pruebas.**

**CAPA DE NEGOCIO:**

**Metodo consultarDisponibilidad en clase GestionHoteles:**

public List<Hotel> consultarDisponibilidad(String nombreHotel, String localidad)

Encuentro estos casos a probar:

1. Búsqueda de un hotel por nombre correcto
2. Búsqueda de un hotel por nombre que no existe
3. Búsqueda por una localidad con más de un hotel
4. Búsqueda por una localidad sin hoteles
5. Búsqueda por una localidad que no existe

**h = new Hotel("NombreValido", "Barrio Padierne", "Santander");**

**h2 = new Hotel("NombreValido2", "Barrio Padierne2", "Santander2");**

**Tabla 1. Casos de prueba unitarias de consultarDisponibilidad().**

|  |  |  |
| --- | --- | --- |
| Entrada | | Salida |
| “NombreValido” | [h] | |
| “NombreInexistente” | [] | |
| “LocalidadConHoteles” | [h, h2] | |
| “LocalidadSinHoteles” | [] | |
| “LocalidadInexistente” | [] | |

**CAPA DE PERSISTENCIA (DAO):**

public Hotel getHotel(String nombre) {

// **TODO** Auto-generated method stub

return em.find(Hotel.class, nombre);

}

Encuentro estos casos a probar:

1. Busco por un nombre de un hotel existente.
2. Busco por un nombre de un hotel que no existe.

**Tabla 2. Casos de prueba unitarias de getHotel().**

|  |  |  |
| --- | --- | --- |
| Entrada | | Salida |
| Hotel h:  Nombre:“Hotel Existente” | h | |
| Hotel h:  Nombre:“Hotel IExistente” | null | |

**PRACTICA 3.2 – CAPA DE NEGOCIO**

Se implementa la capa de negocio de la aplicación (UCHotelesBusiness y UCHotelesCommon) y las pruebas unitarias en GestionHotelesTest.java.

**PRACTICA 3.3 – CAPA DE PERSISTENCIA**

**Diseño de la base de datos:**

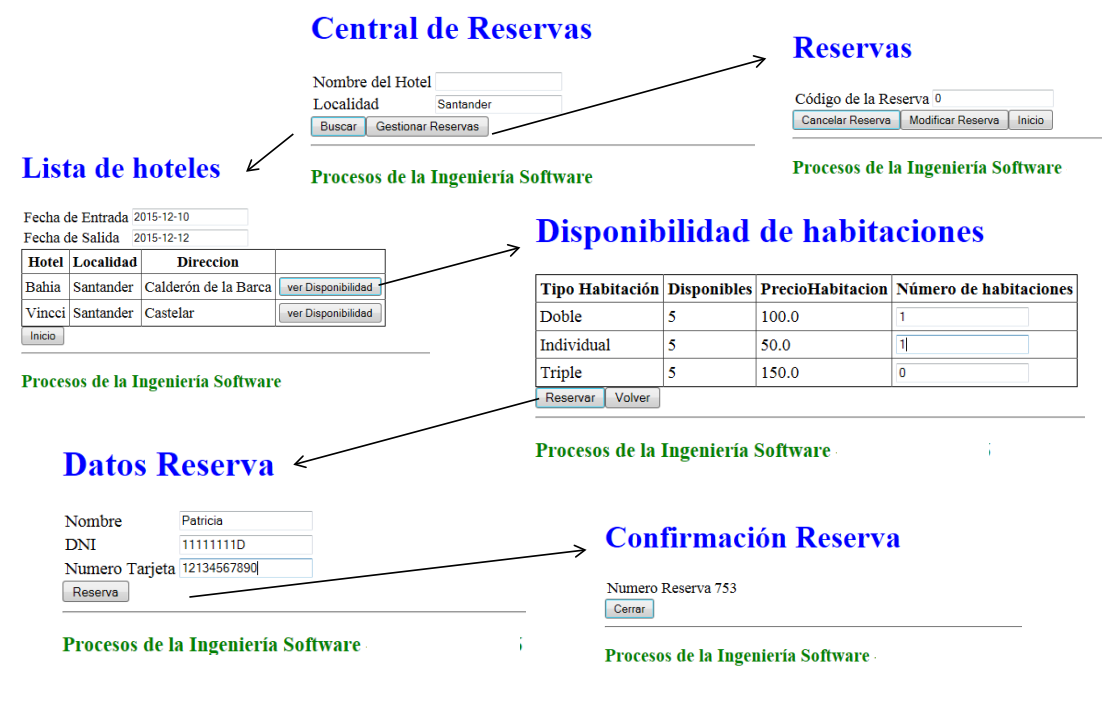
Las relaciones representan las foreign keys, por ejemplo, en la tabla TipoHabitacion se añadiría un campo hotel\_habitaciones\_fk con el nombre del hotel. El enumerado tipo tarjeta es incluido en la tabla Tarjetas con la etiqueta: *@Enumerated*(*EnumType*.***STRING***)

**A computer screen shot of a diagram

Description automatically generated**

También se implementa el código de la capa DAO y las pruebas unitarias de la capa DAO en HotelesDAOTest.java.

**PRACTICA 3.4 – CAPA WEB**



Las vistas (facelets) que forman la capa web son:

1. Central de reservas
2. Lista de hoteles
   1. Validar que fechaSalida > fechaEntrada
3. Disponibilidad de habitaciones
4. Datos reserva
   1. Validar DNI y Numero Tarjeta
5. Confirmación reserva

Será necesario transformar algunos datos recibidos en la vista de String a Date o Int para utilizarlos en las llamadas a métodos de la capa de negocio.

La navegación entre las vistas está definida en la imagen.

Como solo vamos a implementar la parte relacionada con el caso de uso Reservar, usaré un solo Backing Bean que tendrá todos los métodos correspondientes a este caso de uso.