# Enunciado Práctica Java EE
# Iteración 1: aplicación web con JSF

SCS, 2011/12

11 de octubre de 2011

## Índice



## 1. Descripción del problema

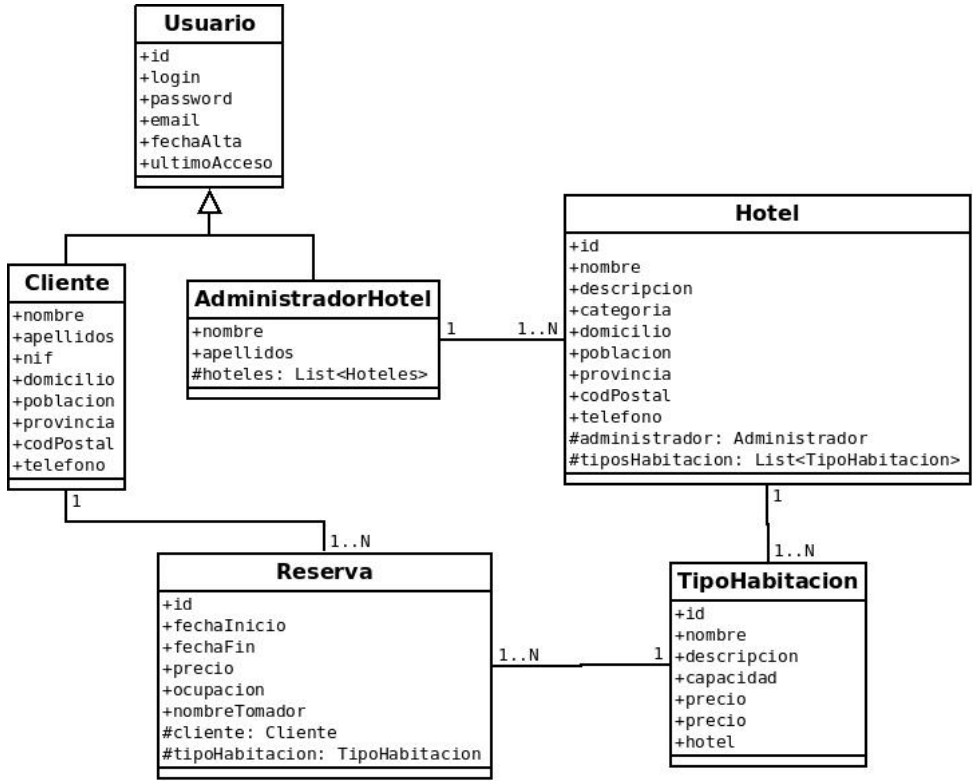
En esta primera iteración de la práctica en grupos se tratará de implementar un pequeño projecto con Java EE para la gestión de una central de reservas hoteleras muy simplificada.

Los objetivos de la práctica son:

- Desarrollar una aplicación distribuida medianamente compleja con una estructura de tres capas (*3-tier*) de tipo cliente ligero: capa de datos, capa de negocio y capa de presentación WEB.
- Manejar y desarrollar componentes Java EE del lado de servidor para implementar la funcionalidades requeridas:
  - La capa de negocio de la aplicación se desarrollará empleando *Enterprise JavaBeans (EJB)* y *Java Persistence API (JPA)*.
  - La capa de presentación se desarrollará empleando la tecnología *Java Server Faces (JSF)*

**Importante:** Se recomienda tomar como muestra la aplicación JEE de muestra *EjemploPedidos* disponible en la web de la asignatura

El modelo de datos de la *central de reservas* a implementar se muestra en el siguiente diagrama de clases, junto con los campos de las tablas MySQL que lo implementan.




**Tablas de la base de datos que mapean las clases y objetos.**

```
Tabla HOTEL:   ID, NOMBRE, DESCRIPCION, CATEGORIA,
               DOMICILIO, LOCALIDAD, CODPOSTAL, PROVINCIA, TELEFONO,
               ADMINISTRADOR_ID, VERSION

Tabla TIPOHABITACION:  ID, NOMBRE, DESCRIPCION,
                       CAPACIDAD, PRECIO, CANTIDAD
                       HOTEL_ID, VERSION

Tabla USUARIO:  ID, TIPO_USUARIO,
                LOGIN, PASSWORD, EMAIL, FECHAALTA, ULTIMOACCESO,
                NOMBRE, APELLIDOS, NIF, DOMICILIO, LOCALIDAD,
                CODPOSTAL, PROVINCIA, TELEFONO, VERSION
        (cada tupla almacena datos de Clientes o de AdministradoresHotel
         se distinguen por el capo STRING TIPO_USUARIO ["cliente" ó "administrador"])

Tabla RESERVA: ID, CANTIDAD, FECHAINICIO, FECHAFIN,
               NOMBRETOMADOR, OCUPACION, PRECIO,
               TIPOHABITACION_ID, CLIENT_ID, VERSION
```

Se han supuesto las siguientes simplificaciones:

- Cada *Hotel* se describe por sus caractarísticas básicas (nombre, dirección, categoría) y ofrece un conjunto fijo de *Tipos de Habitaciones* (entidad débil).
  - Cada *Tipo de Habitación* tendrá sus propias características, las relevantes para nuestro sistema de gestion de reservas son:
    - número de habitaciones disponibles de cada tipo
    - capacidad máxima de dichas habitaciones (capacidad)
    - precio por noche, que por simplicidad se supondrá único (sin promociones o tarifas distintas para temporada alta o baja)
- Para los *Clientes* de la central de reservas se podrán realizar sus reservas en cualquiera de los hoteles que tengan disponibilidad en las fechas que correspondan
  - Cada *Cliente* tiene sus datos personales (nombre, NIF, domiclio, etc)
  - Para realizar una *Reserva* para un *Cliente* este deberá haberse dado de alta empleando el interfaz WEB de la aplicación.
  - Cada *Cliente* registrado tiene accesos a las *Reservas* que haya realizado, permitiéndosele consultarlas y modificar y/o cancelar las que aún no hayan tenido lugar (fecha de inicio posterior a la fecha actual)
- El proceso de confeccionar una *Reserva* por parte de un *Cliente* registrado se organizará en tres fases
  1. Selección del *Hotel* (búsqueda por localidad o nombre de Hotel)
  2. Comprobación de la disponibilidad de habitaciones de la capacidad deseada en las fechas indicadas
  3. Selección del *Tipo de Habitación* y confección de la reserva
     - Para cada *Reserva* se toma nota de:
       - *Cliente* que la realiza
       - *Tipo de Habitación* reservada (e implicitamente el *Hotel* al que pertenece)
       - número de ocupantes efectivos (ocupación)
       - fechas de entrada y salida
       - nombre de la persona q nombre de quien queda hecha la Reserva (por defecto será el nombre del Cliente)
       - importe por noche (por defecto se toma el importe asociado al tipo de habitación, no se consideran promociones o descuentos)

- Cada *Hotel* tiene un *AdministradorHotel* que tendrá capacidad para modificar los datos generales del *Hotel* que administra, dar de alta nuevos *Hoteles* en el sistema y gestionar los *TipoHabitacion* ofertados por cada uno de los hoteles bajo su responsabilidad.

  Asímismo, el *AdministradorHotel* tiene acceso a las *Reservas* realizadas en sus *Hoteles*, para consultarlas, modificarlas o aliminarlas.

- Desde el punto de la capa de presentación WEB tanto los *Clientes* como los *AdministradorHotels* son ambos *Usuarios* y acceden a la aplicación WEB mediante un *login* y una *contraseña* (ver modelo de clases).

  - En el interfaz WEB se distinguen las distintas fuincionalidades disponibles a cada tipo de *Usuario*, ofreciendo distintos formularios a *Clientes* y *AdministradoresHotel*.

# 2. Especificaciones

Dada la base de datos MySQL de partida se deberá implementar una capa de negocio y un ejemplo de cliente ligero WEB para comprobar las funcionalidades de la aplicación.

En el código de partida se aporta un proyecto *NetBeans* de tipo *Enterprise Application*, de nombre **GestionReservas**, con dos módulos:

- **GestionReservas-ejb**: Subproyecto de tipo *EJB Module* donde ya se cuenta con las Entidades JPA para implementar el acceso a la base de datos y donde se deberán incorporar los EJBs necesarios para satisfacer los casos de uso descritos a continuacón.
- **GestionReservas-war**: Subproyecto de tipo *Java Web Application* donde se implementarán una serie de páginas JSF (*Java Server Faces*), junto con sus correspondientes *Managed Beans* de sesión que serán quienes realicen las llamadas a esos EJBs.

**Nota:** Finalmente no se considerará la implementación de aplicaciones de escrito que actúen como clientes accediendo a los EJB mediante llamadas RMI/IIOP.

## 2.1. Capa de negocio: Entidades JPA

Se parte del siguiente conjunto de Entidades JPA, disponibles en el paquete `entidades` del proyecto *GestionReservas-ejb*, que mapean la base de datos de la aplicación.

**Hotel.** Mapea la tabla *Hotel*, tiene una relación 1:N (*one-to-many*) con *TipoHabitaciones*.

En dicha relación todas la operaciones (modificaciones, borrados, ...) sobre un *Hotel* se aplican en cascada sobre sus *Tipos de Habitación* (`cascade=CascadeType.ALL`)

La carga de los *TipoHabitacion* vinculados a un *Hotel* se hace automáticamente (`fetch=FetchType.EAGER`).

**TipoHabitación.** Mapea la entidad débil *TipoHabitación* dependiente de Hotel, tiene una relación N:1 (*many-to-one*) con *Hotel*

**Cliente.** Mapea la tabla *Cliente*.

**AdministradorHotel.** Mapea la tabla *AdministradorHotel*.

**Usuario.** Mapea la tabla *Usuario* (es la superclase de *Cliente* y *AdministradorHotel*).

**Reserva.** Mapea la tabla de *Reservas*. Tiene una relación N:1 (*many-to-one*) con *TipoHabitaciones* y otra relación N:1 con *Cliente*.

## 2.2. Capa de negocio: Enterprise JavaBeans

El grueso de la práctica supondrá el desarrollo de un conjunto de *Enterprise JavaBeans(EJBs)* que den soporte al los *"casos de uso"* descritos en esta sección. Dichos EJBs se encargarán de ofrecer a los cliente una *"fachada"* de la capa de negocio con dos grandes tipos de servicios:

- Mantenimiento básico de las Entidades del sistema: altas, bajas y modificaciones de Clientes, Hoteles (junto con sus Tipos de Habitación) y Reservas
- Consulta de disponibilidad de habitaciones y confección de reservas para Clientes

### 2.2.1. CASO de USO 1: Mantenimiento de Clientes

Se ofrecerá a las aplicaciones clientes las siguientes funcionalidades:

- Búsqueda de Clientes por ID de Cliente: devuelve un único Cliente
- Búsqueda de Clientes por Nombre: devuelve un único Cliente (el primero cuyo nombre coincida con el argumento indicado)
- Búsqueda de todos los Clientes: devuelve una lista de Clientes (`List<Cliente>`)
- Creación (alta) de nuevos Clientes (incluyendo su *login* y *password*)
- Modificación de los datos de un Cliente ya existente (incluyendo su *login* y *password*)
- Eliminación de un Cliente ya existente

### 2.2.2. CASO de USO 2: Mantenimiento de Hoteles

Se ofrecerá a las aplicaciones clientes las siguientes funcionalidades:

- Búsqueda de Hoteles por ID de Hotel: devuelve un único Hotel, que irá acompañado de su lista de Tipos de Habitaciones (en su atributo `tiposHabitacion`)
- Búsqueda de Hoteles por Nombre: devuelve un único Hotel, que irá acompañado de su lista de Tipos de Habitaciones (en su atributo `tiposHabitacion`)
- Búsqueda de todos los Hoteles: devuelve una lista de Hoteles (`List<Hotel>`) acompañados de sus respectivas lista de Tipos de Habitaciones
- Búsqueda de Hoteles por Localidad: devuelve una lista de Hoteles, `List<Hotel>`
- Creación (alta) de nuevos Hoteles

  Hay dos opciones para implementar el alta de Hoteles

  1. Dar de alta únicamente los datos los datos del Hotel, dejando que la adición de sus Tipos de Habitación se gestione como una modificación más
  2. Incluir la lista de Tipos de Habitación (`setTiposHabitacion(...)` entre los datos a dar de alta (el *EntityMannager* hará sobre ellos un `merge()` en cascada)
- Modificación de los datos de un Hotel ya existente (incluida la adición/modificación/eliminación de Tipos de Habitación)
- Eliminación de un Hotel

**Nota:** La relación 1:N (*OneToMany*) entre Hotel (entidad principal) y Tipo de Habitación (entidad débil) está anotada como `cascade=ALL`, de forma que todas las operaciones del *EntityManager* de JPA (`persist()` [creación], `merge() [actualización]` y `remove()` [borrado]) se apliquen en cascada desde Hotel hacia sus Tipos de Habitación, por lo que no es necesario un procesamiento especial en estos casos.

### 2.2.3. CASO de USO 3: Mantenimiento de Reservas

Se ofrecerá a las aplicaciones clientes las siguientes funcionalidades:

- Búsqueda de Reservas por ID de Hotel y rango de fechas: devuelve una lista de Reservas, `List<Reservas>`
- Búsqueda de Reservas por ID de Cliente y rango de fechas: devuelve una lista de Reservas, `List<Reservas>`
- Búsqueda de Reservas por ID de Reserva: devuelve una única Reserva
- Búsqueda de todas las Reservas de un Hotel: devuelve una lista de Reservas (`List<Reserva>`) dado un ID de Hotel
- Búsqueda de todas las Reservas de un Cliente: devuelve una lista de Reservas (`List<Reserva>`) dado un ID de Cliente
- Creación (alta) de Reservas
- Modificación de los datos de una Reserva
- Eliminación de una Reserva

### 2.2.4. CASO de USO 4: Control de Disponibilidad

Implementa los procesos de negocio relacionados con la búsqueda de Hoteles, las consultas de disponiblidad y la confección de Reservas para un Cliente.

Se ofrecerán a las aplicaciones clientes las siguientes funcionalidades:

- Búsqueda de Hoteles por Localidad: devuelve una lista de Hoteles, `List<Hotel>`
- Consulta de la disponibilidad de un Hotel (dado su Número de Hotel) entre unas fechas dadas
  - Devolverá una lista de objetos de la clase *Disponibilidad* (`List<Disponibilidad>`)
    - Esta clase auxiliar está definida en el paquete `auxiliar` del proyecto
    - Almacena dos atributos
      - Un objeto Tipo de Habitacion
      - Un entero con el número de habitaciones de ese tipo disponibles
    - Es una clase serializable para que pueda ser enviada entre los clientes y el EJB
  - Como punto de partida se puede usar la consulta JPQL del siguiente ejemplo.

```
Query q = em.createQuery("SELECT sum(r.cantidad) FROM Reserva r " +
             " WHERE r.tipoHabitacion.id = :tipo " +
             " AND   ((r.fechaInicio BETWEEN :inicio AND :fin) OR " +
             "        (r.fechaFin BETWEEN :inicio AND :fin) OR " +
             "        (r.fechaInicio <= :inicio AND r.fechaFin >= :fin)) ");
```

    - Recibe 3 parámetros:
      - ID del Tipo de Habitación consultado (`:tipoHabitacion`)

    - ◇ Fecha de inicio (`:inicio`), de tipo *Date*
    - ◇ Fecha de fin (`:fin`), de tipo *Date*
  - Devuelve, para un Tipo de Habitación dado, un objeto *Integer* con el número de habitaciones ocupadas entre las fechas indicadas.

Para crear la lista de objetos *Disponibilidad* se recorre la lista de Tipos de Habitación de un Hotel, restando a la cantidad de habitaciones disponibles de cada Tipo de Habitación la cantidad devuelta por la consulta.

```
...
Hotel hotel = em.find(Hotel.class, numHotel);

List<Disponibilidad> lista = new Vector<Disponibilidad>();
for (TipoHabitacion tipo : hotel.getTiposHabitacion()) {
     q.setParameter("tipo", tipo.getId());
     q.setParameter("inicio", fechaInicio);
     q.setParameter("fin", fechaFin);
     ocupadas = (Long) q.getSingleResult();

     if (ocupadas == null) {
         libres = tipo.getCantidad();
     }
     else {
         libres = tipo.getCantidad() - ocupadas.intValue();
     }
     lista.add(new Disponibilidad(tipo, libres));
}
return(lista);
...
```

*numHotel*, *fechaInicio* y *fechaFin* serían los parámetros del método remoto

**Nota:** La creación de una Reserva para un Cliente, un Tipo de Habitación y unas fechas dadas es gestionada por el EJB encargado del *"caso de uso 3"*

## 2.3. Capa de presentación WEB

La capa de presentación WEB se implementará empleando JSF (*Java Server Faces*). Se aprtirá del proyecto *GestionReservas-war* donde se proporciona un esqueleto de partida en el cual las tareas de autenticación de ambos tipos de *Usuario* (*Cliente* y *AdministradorHotel*) ya están implementadas. Se incluye también el formulario de alta de nuevos *Clientes*.

La aplicación WEB a desarrollar ofrecera dos conjuntos de funcionalidades:

- las destinadas a los *Clientes*: búqueda de Hoteles/Habitaciones, confección de Reservas y gestión de sus Reservas
- las destinadas a los *AdministradoresHotel*: alta y modificación de sus Hoteles, consulta y gestión de sus Reservas

El desarrollo de la capa de Presentación WEB, junto con los elementos de la capa de Negocio (EJBs) necesarios en cada caso se repartirá del siguiente modo:

#### 2.3.1. Funcionalidades para los ADMINISTRADORES de HOTELES de la central de reservas

Deberán implementar las páginas JSF (junto con los EJB ["casos de uso" 1 a 4] que estas precisen) para dar soporte a :

1. Listado de los Hoteles gestionados
2. Mantenimiento de los datos de los Hoteles y sus Tipos de Habitación
   - Altas de nuevos Hoteles (junto con sus Tipos de Habitación)
   - Modificación de Hoteles (junto con alta de nuevos Tipos de Habitación o su modificación)
3. Listado y gestión de las Reservas del Hotel actual (uno de los gestionados por este AdministradorHotel). Permitirá:
   - Ver las todas las Reservas (anteriores y actuales) del Hotel
   - Modificar Reservas aún no iniciadas (fecha de inicio posterior a fecha actual)
   - Cancelar Reservas aún no iniciadas (fecha de inicio posterior a fecha actual)

La página JSF a partir de la cual trabajar será **gestionAdministrador.xhtml**

#### 2.3.2. Funcionalidades para los CLIENTES de la central de reservas

Deberán implementar las páginas JSF (junto con los EJB ["casos de uso" 1 a 4] que estas precisen) para dar soporte a :

1. Listado y gestión de las Reservas del Cliente actual. Permitirá:
   - Ver las todas las Reservas (anteriores y actuales) del Cliente
   - Modificar Reservas aún no iniciadas (fecha de inicio posterior a fecha actual)
   - Cancelar Reservas aún no iniciadas (fecha de inicio posterior a fecha actual)
2. Confección de una nueva Reserva, conforme a los sguientes pasos
   - Búsqueda y presentación de Hoteles (junto con sus Tipos de Habitación) en una Localidad
   - Selección de un Hotel y comprobación de Disponibilidad en base a Fecha de Entrada, Fecha de Salida y Número de Ocupantes, mostrando la lista de Tipos de Habitación disponibles
   - Selección de Tipo de Habitación, confección de la Reserva y confirmación.

   **Nota:** Si se prefiere, podrá seguirse otro esquema más directo en la búsqueda, comprobación de disponibilidad y confección de la Reserva (por ejemplo: indicar directamente en la primera búsqueda la fechas de inicio y fin y el número de ocupantes)

La página JSF a partir de la cual trabajar será **gestionCliente.xhtml**

## 3. Tareas y documentación a entregar

La práctica se puede hacer de forma individual o en grupos de 2 a 4 alumnos. Las tareas a realizar serán distintas en función del tamaño del grupo. En caso de duda consultar con el profesor (`ribadas@uvigo.es`).

**Fecha límite de entrega:** Como último día se fija el **Lunes 23/1/2011** (despacho 303)

## 3.1. Tareas concretas

Funcionalidades mínimas en función del tamaño del grupo.

**Grupos de 4 miembros:**

- Funcionalidades de los clientes de la central de reservas
  - Dar soporte a todas las funcionalidades incluidas en la sección 2.3.2
- Funcionalidades de los administradores de la central de reservas
  - Dar soporte a todas las funcionalidades incluidas en la sección 2.3.1

## 3.2. Documentación para la iteración 1

Se deberá entregar en CD/DVD (o en una memoria USB desde la que hacer la copia) lo siguiente:

- Directorio del proyecto NetBeans completo, con el código fuente de la implementación desarrollada
- Fichero EAR resultante de compilar y empaquetar el proyecto completo

Se entregará una memoria breve en papel con la siguiente estructura (aprox. 5-6 páginas):

1. Descripción breve de la aplicación, indicando los componentes de los que se partía y los componentes implementados

   **Nota:** incluir los nombres, DNI y e-mail de los miembros del grupo en la portada
2. Descripción de la capa de negocio implementada
   - Describir la estructura de la solución implementada (tanto para la parte de gestión de clientes como para la de gestión de administradores)
   - Detallar los interfaces de negocio definidos
     Para cada uno, indicar su nombre, su tipo (local o remoto), la lista de métodos y sus argumentos, etc
   - Detallar los EJB que implementan esos interfaces de negocio.
     Para cada uno, indicar y justificar su tipo (*stateless* o *stateful*), comentar los detalles que sean relevantes de la implementación de sus métodos (consultas JPQL definidas, funcionalidades que ofrece, etc)
3. Descripción de la capa de presentación WEB
   - Detallar la estructura de la solución implementada
   - Describir las funcionalidades de las páginas JSF que componen la capa de presentación (tanto para la parte de gestión de clientes como para la de gestión de administradores)
     - Enumerar y explicar brevemente las páginas JSF creadas, sus funcionalidades y el fujo entre ellas.
   - Describir los *managed beans* empleados: sus atributos y acciones y su interacción con los EJBs de la capa de negocio.
4. Conclusiones, problemas/dificultades encontrados, comentarios, etc

**Importante:** No es requsito imprescindible para que la práctica sea evaluada que la aplicación se llegue a ejecutar en el servidor de aplicaciones. Se podrá entregar una práctica *"no ejecutable"* a condición de que se hayan implementado la totalidad de los componentes previstos en la especificación y estos se documenten convenientemente.