PRACTICA FINAL 2011/2012 : "Reserva de hoteles"

El problema que se presenta en la práctica es la implementación de una aplicación web para la oferta y reserva de hoteles. A través de la aplicación, los usuarios podrán reservar plazas hoteleras, cancelar reservas, dejar un comentario sobre la reserva y listar los hoteles .

Además, el sistema identificará a un usuario administrador que permitirá realizar tareas específicas de administración como la introducción de hoteles y plazas hoteleras, ofertar descuentos o gestionar la no disponibilidad temporal de las plazas.

Nota: Este enunciado toma como guía a la aplicación web **Booking.com** (http://www.booking.com/). Se trata de una versión muy simplificada de la misma, pero ante cualquier duda se puede consultar el funcionamiento actual de dicha aplicación web.

Enunciado

La aplicación 'Reserva de hoteles' tiene como principal propósito ofrecer una herramienta telemática para gestionar la oferta y reserva de plazas hoteleras.

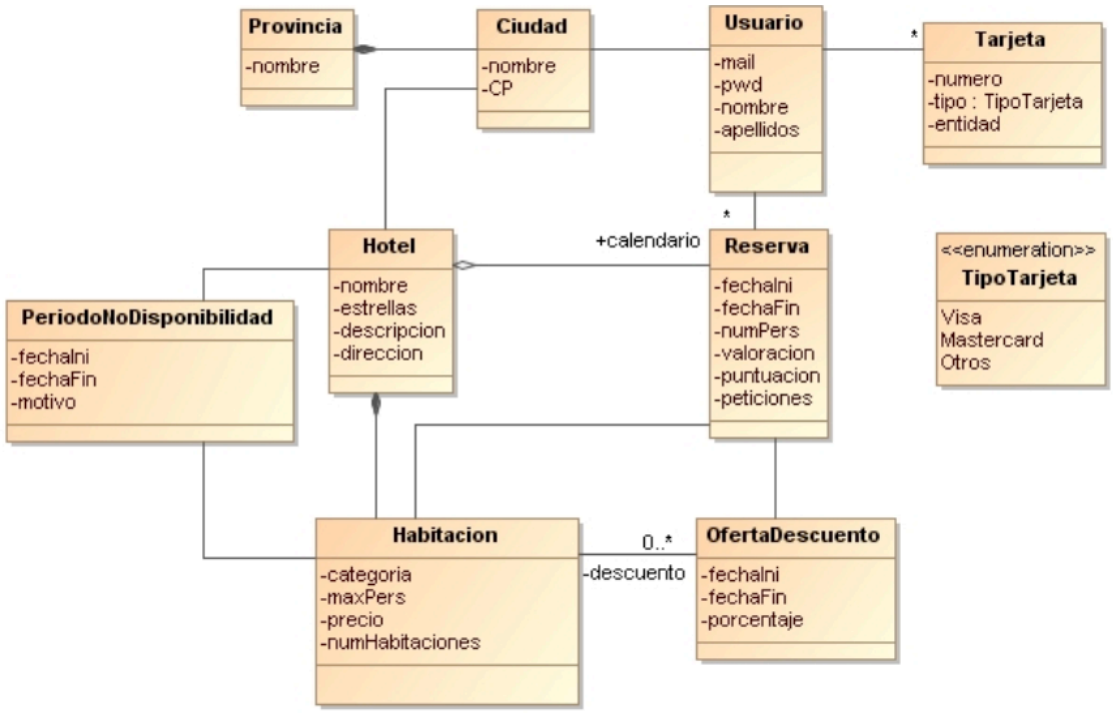
Los hoteles (administrador) podrán registrar su información y añadir la cantidad de habitaciones que ofertan, clasificadas por categoría (el tipo de habitación puede ser: doble, simple, triple, suite, etc…). Para cada categoría de habitaciones indicarán un precio y el numero máximo de personas que puede alojar cada habitación.

Los usuarios podrán registrase en la aplicación, introduciendo el mail, password, nombre y apellidos. Opcionalmente podrán indicar la ciudad donde residen. Una vez que el usuario está registrado podrá realizar reservas de habitaciones. Para ello, primero listará las habitaciones disponibles de acuerdo a unos criterios de búsqueda. El usuario podrá reservar entonces las habitaciones desde el listado. Para reservar el sistema tomará las fechas de entrada y salida establecidas en los criterios de búsqueda. Además el usuario deberá introducir para las habitaciones de hotel seleccionadas, el número de personas por habitación y unos comentarios con peticiones adicionales de la reserva.

Los usuarios también podrán cancelar las reservas y comentarlas (no se considerarán restricciones de fecha ni para la cancelación ni para los comentarios).

Por último, los hoteles podrán establecer periodos de fechas en las que un tipo de habitaciones o incluso el hotel completo no estará disponible (por simplicidad no se considerarán habitaciones en particular si no tipos de habitación a la hora de establecer periodos de no disponibilidad). De igual modo, los hoteles podrán establecer periodos de descuentos sobre tipos de habitaciones, indicando el porcentaje de descuento sobre el precio de la habitación.

A continuación se muestra un modelo conceptual muy básico sobre el problema (considerar atributos descritos en los requisitos del enunciado e incluso algún otro atributo que el alumno considere oportuno):



Este modelo conceptual comprende todas las clases que a posteriori deben implementarse. Pero tan solo se muestran los atributos básicos de las clases, resultando necesaria la adición de nuevos atributos. Añadir conforme se vayan necesitando.

Los casos de uso identificados en el enunciado son los siguientes (se indicará una breve descripción textual de manera informal):

- **(U) Registrar usuario (incluye tarjetas)**

El usuario registrará los siguientes datos: mail (que servirá como identificador), password, nombre, apellidos y ciudad donde reside. El sistema generará una url para la activación que debería ser enviada por mail al usuario para la activación de su cuenta de usuario, pero que en el caso de la práctica que nos ocupa, será suficiente con mostrar una la url en el formulario de registro para que se realice la activación desde el mismo.

- **(U) Reservar habitación**

El usuario podrá realizar reservas desde el listado obtenido por la búsqueda de habitaciones (ver cdu más abajo). El sistema mostrará un formulario con los datos de la reserva. Se mostrarán los datos de la habitación, el hotel y el precio final incluido descuento y número habitaciones (datos no editables). El formulario de edición mostrará las fechas de entrada y salida (tomadas del listado, pero

modificables, requiriendo en ese caso validación de disponibilidad), el número de personas por habitación, el número de habitaciones (por defecto 1) y peticiones (a modo de comentario) de la reserva. El sistema registrará la reserva en las fechas indicadas y con los datos introducidos.

- **(U) Cancelar reserva**

El sistema mostrará la lista completa de reservas del usuario (ya ocupadas y pendientes de ocupación) ordenadas por fecha de entrada (primero la mas reciente). El usuario marcará en el listado (mediante casilla de marcado) las reservas a cancelar. El sistema pedirá confirmación antes de proceder al borrado lógico de la reserva (la reserva quedará marcada como cancelada, pero no borrada físicamente del historial de reservas).

- **(U) Comentar reserva**

El sistema mostrará la lista de reservas ya ocupadas del usuario (solo las ocupadas). El usuario seleccionará una reserva. El sistema mostrará los datos de la reserva (mismos datos que en el cdu reservar, pero en este caso ninguno es editable). El usuario introducirá los comentarios sobre su estancia y establecerá una valoración entre 0 y 10 (solo valores enteros). El sistema registrará los comentarios y la valoración final.

- **(U) Listar hoteles**

El sistema listará las habitaciones disponibles de acuerdo a un criterio de búsqueda: nombre de la ciudad o provincia u hotel (obligatorio alguno de los tres), fecha de entrada y salida (obligatorio), y número de personas a ser alojadas (este último opcional).
El listado podrá ser ordenado por: nombre del hotel, valoración media del hotel, precio final incluidos descuentos (mayor o menor) y/o ciudad. Por defecto aparecerá la cantidad de 1 habitación a reservar, pudiendo el usuario concretar el número de habitaciones de cada categoría.
El listado podrá ser agrupado por hotel ó ciudad.

Nota: Consultar los listados de **Booking.com** para disponer una organización de la información similar (considerando evidentemente la reducción de información que tiene esta práctica)

- **(A) Registrar datos de hotel (incluye habitaciones)**

El administrador introducirá los datos del hotel: nombre, número de estrellas, descripción, dirección y ciudad de ubicación. El administrador dispondrá también de otro formulario para introducir las habitaciones del hotel: categoría (tipo de habitación), cantidad de habitaciones, máximo número de personas permitidas, precio. El administrador introducirá tantas veces como desee la anterior información de las habitaciones. Véase que el sistema debería comprobar que sólo se puede introducir una vez la información de habitaciones de un tipo (categoría).

El administrador da por finalizada la introducción de las habitaciones y el sistema registra los datos del hotel.

- **(A) Establecer oferta**

El administrador indica un periodo (fecha inicio y fin) para aplicar un descuento sobre un tipo (categoría) de habitaciones de un hotel. El administrador indica el descuento en términos porcentuales. El sistema registra el periodo de descuento.

- **(A) Establecer no disponibilidad**

El administrador indica un periodo (fecha inicio y fin) de no disponibilidad sobre un hotel o un tipo de habitación de un hotel. El administrador indica motivo de la no disponibilidad. El sistema comprueba que dicho periodo no afecta a ninguna reserva en pendiente. El sistema registra el periodo de no disponibilidad.

A considerar

- Realizar solo el registro de usuarios. No es necesario el borrado o la modificación.
- Se considera que existe un administrador que se encarga de registrar toda la información de los hoteles (no hay usuario 'hotel')
- Puede haber más de un descuento sobre un tipo de habitación.
- El sistema no registra las habitaciones de manera individual, si no por tipos de habitación, indicando la cantidad de habitaciones que el hotel tiene de ese tipo (categoría).
- Los listados de habitaciones por defecto sólo muestran habitaciones disponibles (no muestra aquellas que están ya ocupadas o marcadas como no disponibles). Mediante una cailla de marcado se puede indicar que se incluyan las habitaciones no disponibles, indicando el motivo de la no disponibilidad (reservada o un motivo particular indicado a nivel de periodo de no disponibilidad).
- Las ciudades y provincias estarán insertadas previamente en la base de datos.

Sobre los usuarios y las opciones de la aplicación

La aplicación considerará un tipo especial de usuario: 'administrador', cuyo nombre de ***usuario*** y ***clave*** están identificados a nivel de aplicación en el fichero de configuración web, con valores 'admin' y 'admin', además de estar registrado en el sistema (la aplicación lo registrará automáticamente la primera vez que se realice el login con 'admin/admin').

Implementación

Las prácticas serán realizadas **en grupos de dos alumnos.**

La práctica deberá ser desarrollada en un total de **dos** entregas cuyas modificaciones paulatinas son acumulables. *[Se recuerda al alumno que los boletines que se implementan en el laboratorio no son entregables]*

Las entregas están organizadas del siguiente modo:

- 1ª Entrega: **Fecha: 23/12/2011**
- 2ª Entrega: **Fecha: 16/01/2012**

Requisitos Entregable 1 → Servlets, Persistencia, Vista, MVC

El alumno debe implementar la función de 'Login' mediante el uso de Servlets.

La opción de 'Login' debe permitir la identificación del usuario en la aplicación. El proceso de login debe admitir un máximo de 3 intentos, no permitiendo un 4º intento en la misma sesión. Además, el sistema debe llevar un fichero de log ('acceso.log') donde se registren todos los sucesos ocurridos durante el logueo de cualquier usuario. **Esta funcionalidad debe implementarse mediante un Servlet (ServletLog).**

Todas las vistas estarán implementadas con JSP o JSF. En MVC, las clases acción delegarán en un/os controlador/es.

Requisitos minimos:

- **Servlets (ServletLogin)**
- **Uso de una tecnología de persistencia**
- **Vistas implementadas con una tecnología de presentación: JSP haciendo uso de MVC o JSF**

Los requisitos mínimos puntúan **6 sobre 10**. Para alcanzar el 10 en la entrega, es necesario implementar haciendo uso de más tecnologías, de entre las vistas en clase. Como guía indicar que cada tecnología adicional aplicada se valorará con **+2** puntos aproximadamente (tal vez un poco mas de dos o un poco menos, dependiendo de la complejidad de la misma). Estas puntuaciones se indican suponiendo una aplicación correcta y completa de la tecnología.

Implementaciones que se pueden aplicar en la primera entrega:

- Generación de interfaces dinámicas: **HTML Dinámico** y **AJAX**
- Componentes de servidor para la generación de páginas dinámicamente: **Servlet**, **JSP** y **JSF** (considerar el uso de ***JSTL***, JSP en diferentes versiones, etc…)
- Persistencia: **JDBC**, **Hibernate, JPA** (considerar la aplicación del patrón ***DAO***)
- Patrón arquitectónico **MVC**

Para el listado de datos, se puede hacer uso ***puntual*** de la librería SQL de JSTL.

No hay que duplicar la implementación de funcionalidades. Se pueden implementar ciertas funcionalidades con unas tecnologías y otras funcionalidades con otras tecnologías.

→ Esquema/Catálogo 'AADD', usuario 'root', password ''

Entregable 2 → RMI, EJB y Patrones de la capa de negocio

Realizar una segunda versión de la práctica donde:

- Al menos 1 **controlador** sea EJB de Sesión
- Al menos 3 **objetos de negocio** distribuidos sean EJB.
    - ***Se puede utilizar EJB 2 o EJB 3***
    - ***En EJB 3.0 usar características propias (declaración del vean, inyección de dependencias, etc…)***
    - ***En EJB 2.0, Se puede utilizar persistencia BMP (con la implementación de la persistencia de la 1ª entrega) y/o CMP 2.x.***
- Se aplique al menos 1 **patrón de la capa de negocio**.
- Se implemente el enunciado de **RMI** (ver anexo A)

Los requisitos mínimos puntúan **6 sobre 10**. Para alcanzar el 10 en la entrega, es necesario implementar haciendo uso de más tecnologías, de entre las vistas en clase. Como guía indicar que cada tecnología adicional aplicada se valorará con **+2** puntos aproximadamente (tal vez un poco mas de dos o un poco menos, dependiendo de la complejidad de la misma). Estas puntuaciones se indican suponiendo una aplicación correcta y completa de la tecnología.

No hay que duplicar la implementación de funcionalidades. Se pueden implementar ciertas funcionalidades con unas tecnologías y otras funcionalidades con otras tecnologías.

Valoración

Se tendrá en cuenta la calidad de la aplicación implementada en cuanto al **uso correcto y completo *de cualquiera de las tecnologías estudiadas en la asignatura AADD*** (tanto a nivel de practica como de teoría, excluyendo las tecnologías vistas en el tema 1 de introducción). La nota dependerá de la variedad y corrección de tecnologías aplicadas.

La valoración de la práctica según entregables será:

Entregable 1: 50%
Entregable 2: 35%
Documentación: 15%

La práctica será aprobada si funciona bajo los requisitos mínimos solicitados. La nota final de la práctica se establecerá **de acuerdo a las funcionalidades y tecnologías** aplicadas en el desarrollo.

Todas las entregas son **obligatorias**.

IMPORTANTE:

→ Comprobar las entregas y su correcto funcionamiento y despliegue:
- **Si la práctica NO funciona y NO se consigue ejecutar, estará SUSPENSA**

→ Preparar la entrevista de prácticas.
- El profesor realizará una entrevista puntuable a cerca de la implementación de la práctica.

→ NO se corregirán entregas que no incluya el código fuente (ya sea parcial o totalmente).

Entregables

Dejar en **AulaVirtual** un archivo ZIP con el nombre **PracticaFinal.zip** con el siguiente contenido:

PracticaFinal.zip:
- DOC (carpeta)
  - Documentacion.doc
- APP (carpeta)
  - Despliegue (carpeta)
    - Incluir todos los recurso necesarios para el despliegue
  - GestorHoteles.jar
  - GestorHoteles Web.war
  - ServidorRMI (carpeta)
    - Ficheros .bat, aplicación servidor, objeto RMI, etc…

GestorHoteles.jar

Aplicación .jar con los componentes de la capa de negocio (bean de sesión y entidad). Deben incluirse los fuentes .java en la misma ubicación que su correspondiente .class. Estructurado de manera semejante al entregable de la practica EJB del laboratorio. Debe estar lista para su despliegue en el servidor JBoss (comprobad que el archivo funciona correctamente).

**** Incluir en la carpeta 'Despliegue' todos los recursos necesarios para el despliegue**

GestorHotelesWeb.war

Aplicación web empaquetada en archivo **.war** y estructurada según arquitectura MVC. Deben incluirse los fuentes **.java** en la misma ubicación que su correspondiente **.class**. Estructurado de manera semejante a los entregables de las practicas del laboratorio. Debe estar lista para su despliegue en el servidor JBoss (comprobad que el archivo funciona correctamente).

**** Incluir en la carpeta 'Despliegue' todos los recursos necesarios para el despliegue**

ServidorRMI
Carpeta con la aplicación encargada de activar y servir el objeto distribuido RMI. La estructura debe ser la misma que la incluida en el entregable de la practica RMI del laboratorio (incluidos los archivos .**bat**).

Documentación.doc
Fichero **.doc** (válido también en formato **.pdf**) con toda la documentación relativa a la práctica. Seguir el siguiente formato indicado por capítulos:

- **Capitulo 1 : Introducción**

Indicar todas las consideraciones adoptadas a partir del enunciado (por carencia de requisitos o por modificación, en este último caso aportando la debida justificación).

- **Capitulo 2 : Instalación y despliegue**

Indicar todas las consideraciones a tener en cuenta para el despliegue de la aplicación sobre el servidor JBoss y dentro de la plataforma de ejecución. Por ejemplo:

  - DSN creado y ubicación del archivo **.mdb** de Access dentro de los entregables
  - Scripts de creación para MySQL (indicar versión utilizada, drivers, y todos los pasos para crear el origen de datos).
  - Posibles ficheros de configuración de la aplicación. Ficheros de propiedades (no confundir con los ficheros estándar web.xml, ejb-jar.xml o los propios de JBoss, jaws.xml o jboss.xml) definidos para parametrizar la aplicación (si se hubiera parametrizado).
  - Cualquier otro recurso que referencie al despliegue y la instalación de la práctica.

- **Capitulo 3 : Descripción técnica de la implementación**

Describir **QUE** aspectos de la práctica se han realizado, **COMO** tecnológicamente se han implementado y **DONDE** (en que recursos de la práctica).
Esta descripción debe realizarse para todos los requisitos indicados en la última página del enunciado (básicos y opcionales).
Este es el capítulo más importante de la documentación de la práctica.

- **Capitulo 4 : Nuevos aspectos**

Con el mismo tratamiento visto para los requisitos del capitulo 3, describir aquí funcionalidades implementadas que no fueron indicadas en el enunciado de la práctica.

- **Capitulo 5 : Requisitos BASICOS no implementados**

Listar los requisitos básicos no implementados o si se implementaron pero no funcionan correctamente (indicar donde fallan).

- **Capitulo 6 : Otros**

Indicar en este capítulo cualquier otra consideración a hacer notar en la práctica que no sea introducida en ninguno de los anteriores capítulos.

ANEXO A: DISTRIBUCIÓN CON RMI

Se debe implementar un pequeño sistema distribuido que se encargue de **simular el envío de un mail cada vez que un usuario realice una reserva de hotel**. Debe ser implementado con objetos distribuidos RMI que se lanzarán desde un servidor, como vimos en prácticas.

Al finalizar cada reserva, el sistema generará una notificación por mail al cliente. El mail debe contener la información textual completa de la reserva (identificación del cliente, hotel, habitaciones, cantidades, fechas y precio total). **Implementar la simulación del envío de mail utilizando solo objetos distribuidos RMI**

Se creará un ***Gestor de Mails*** para simular el envío de mail. Se trata de un gestor de mails que se implementará con RMI y tendrá los siguientes objetos distribuidos (2 objetos RMI):

- Un **controlador gestor de mails** que ofrece un único método **enviar** que devuelve un booleano indicando si todo fue bien. Además el funcionamiento del gestor es como sigue. El gestor llevará una colección de mails (no persistentes) enviados en la sesión del servidor. El nombre de este objeto en el Servicio de Nombres es '**GestorMails'**. Esta colección no es persistente en BBDD, por lo que cuando el objeto distribuido se libere de memoria, debe borrarse. Cuando la colección alcance los 3 envíos (tamaño de la colección == 3), se volcarán los mensajes en la salida estándar de java (System.out) y se vaciará la colección.

- Un **objeto mail** que representa un mail enviado. No es necesario registrar este objeto en el Servicio de Nombres, pues será tratado únicamente por el gestor de mails.

P.D.: No se debe realizar un envío real de mail. El gestor de mail simulará el envío, creando un objeto distribuido Mail y guardándolo en su colección.