

Portal de reserva de billetes de tren

Índice

1. Condiciones generales	1
2. Introducción	1
3. Modelo de datos	1
4. Funcionalidad mínima de la aplicación	2
4.1. Compra de billete	2
4.2. Consulta y cancelación de billetes comprados	2
4.3. Otros requisitos	3
5. Funcionalidad opcional de la aplicación	3
6. Recomendaciones de diseño	4
7. Formato de entrega	4
8. Criterios de evaluación	5
9. Referencias y material complementario	5
9.1. Estándares	5
9.2. Servlets y JSP	5
9.3. MySQL	6

1. Condiciones generales

- Es necesario aprobar esta práctica para aprobar la asignatura.
- La práctica debe ser entregada estrictamente en el plazo estipulado.
- La práctica debe realizarse y entregarse por parejas. Aquellos alumnos que no tengan pareja, o cuya pareja renuncie a trabajar en la práctica, deben hablar con los profesores de la asignatura. Sólo se admitirán entregas individuales en aquellos casos que hayan sido autorizados previamente por los profesores.
- Las parejas deben ser las mismas que en la práctica de Swing, salvo en situaciones excepcionales (por ejemplo, abandono de la asignatura o falta de colaboración de alguno de los alumnos), que deben ser comunicadas a los profesores con la mayor antelación posible.
- Cada pareja utilizará la misma base de datos que en la práctica de Swing.
- Las entregas deben funcionar en los ordenadores del Depto. de Ingeniería Telemática (sistema operativo Linux) de los laboratorios en que se imparten las clases prácticas, usando las bases de datos proporcionadas.
- Para que la práctica sea evaluada, es imprescindible que sus autores se presenten a la entrevista de evaluación de la misma en la fecha y la hora convenidas. En la medida de lo posible, los profesores intentarán ser flexibles en el establecimiento de citas.
- La calificación de esta práctica se conserva, si el alumno lo desea, para la convocatoria de septiembre. Se entenderá que el alumno no desea conservarla si entrega una nueva versión de la práctica en la convocatoria de septiembre, en cuyo caso se evaluará la nueva versión.
- Las prácticas a entregar en la convocatoria de septiembre serán las mismas que en la convocatoria de febrero.

2. Introducción

En esta práctica se desarrollará un portal Web de compra de billetes de tren por parte de clientes finales. Este portal, permitirá, fundamentalmente, consultar horarios de trenes y disponibilidad de plazas, así como comprar billetes y consultar o cancelar billetes comprados previamente.

3. Modelo de datos

El modelo de datos de la aplicación es prácticamente el mismo que se ha publicado en el enunciado de la práctica global Swing. La única diferencia es que se añaden dos tablas nuevas que permiten almacenar los datos de los usuarios registrados en el sistema, así como asociar billetes a usuarios, cuando estos realizan la compra estando autenticados:

```
CREATE TABLE Usuarios (
    uid INT(3) NOT NULL auto_increment,
    nombre VARCHAR(128) NOT NULL,
    alias VARCHAR(16) NOT NULL,
    password VARCHAR(16) NOT NULL,
    PRIMARY KEY (uid),
    UNIQUE INDEX (alias)
) ENGINE = INNODB;

CREATE TABLE BilletesUsuario (
    IdUsuario INT(3) NOT NULL,
    IdBillete INT(11) NOT NULL,
    PRIMARY KEY (IdUsuario, IdBillete),
    INDEX (IdUsuario),
    FOREIGN KEY (IdUsuario) REFERENCES Usuarios (uid)
        ON UPDATE CASCADE,
    FOREIGN KEY (IdBillete) REFERENCES Billetes (IdBillete)
        ON UPDATE CASCADE
) ENGINE = INNODB;
```

Los campos de la tabla de usuarios son los siguientes:

- `uid`: identificador único del usuario, de uso interno en el sistema.
- `nombre`: nombre y apellidos del usuario.
- `alias`: identificador de usuario que este debe introducir cuando se autentica.
- `password`: contraseña del usuario.

La tabla `BilletesUsuario` relaciona los billetes comprados por un usuario autenticado con dicho usuario. Sus campos son los identificadores de billete y usuario de las tablas `Billetes` y `Usuarios`.

4. Funcionalidad mínima de la aplicación

El sistema debe implementar el proceso de compra de un billete, y el de consulta/cancelación de billetes comprados por el usuario. Para realizar una compra es opcional estar autenticado en el sistema. Para obtener un listado de los billetes comprados es obligatorio que el usuario esté autenticado, y sólo se mostrarán aquellos billetes que hubiese comprado estando autenticado con el mismo identificador de usuario.

La vista principal del sistema presenta un formulario que permite realizar una búsqueda de líneas por estación de origen, estación de destino y fecha. Además, en algún lugar de la página debe presentar información relativa a la autenticación de usuarios: si el usuario está autenticado, presenta su nombre y un enlace a la vista de billetes comprados por el usuario; si no está autenticado, presenta un formulario que le permita autenticarse. Una vez autenticado el usuario, la aplicación volverá a mostrar esta vista principal.

4.1. Compra de billete

Para realizar una compra, el usuario debe rellenar en el formulario de la vista principal las estaciones origen, destino y fecha del viaje. El usuario podrá a continuación seleccionar una línea de entre las mostradas y comprar un billete para ella en la categoría indicada por el usuario. No es necesario permitir que el usuario elija vagón, es suficiente con permitirle elegir categoría y que el sistema asigne un vagón automáticamente.

Para realizar la compra, el usuario debe introducir su nombre y apellidos (si está autenticado, el sistema puede automáticamente rellenar este campo) y su número de tarjeta de crédito. Finalmente, el sistema emitirá el billete y mostrará un resumen con los datos del mismo.

Para simular situaciones en que una tarjeta de crédito no sea válida, el sistema considerará que son válidos todos los números de 16 dígitos excepto aquellos que terminen en 0. En caso de que el número introducido no sea válido, el sistema no puede emitir el billete.

Si el usuario está autenticado en el momento de la compra, se le asociará el billete comprado a su cuenta, para que posteriormente pueda obtener un listado de todos sus billetes y cancelarlos.

4.2. Consulta y cancelación de billetes comprados

Si el usuario está autenticado, la página principal del sistema debe presentar un enlace que le lleve a una página en que se muestren todos los billetes no cancelados que haya asociados a su cuenta de usuarios (esto es, todos los billetes que haya comprado estando autenticado con dicha cuenta).

El sistema debe permitir cancelar cualquier billete del usuario cuya fecha y hora de salida sea posterior en al menos 24 horas a la fecha y hora actual.

4.3. Otros requisitos

La aplicación desarrollada, además de proporcionar la funcionalidad mínima que se pide, debe cumplir los siguientes requisitos:

1. Todos los documentos HTML generados deben ser válidos con respecto a la especificación de XHTML 1.1.
2. Debe utilizar CSS para establecer el estilo de las páginas generadas.
3. Debe organizar las clases en paquetes.
4. Debe permitir que múltiples usuarios realicen operaciones simultáneamente.
5. Todas las URLs utilizadas deben ser relativas, salvo en casos en que esté justificado el uso de una URL absoluta. En la corrección se instalará la práctica con un nombre de contexto distinto, por lo que si las URLs no son totalmente relativas, probablemente la aplicación no funcione.

5. Funcionalidad opcional de la aplicación

Adicionalmente, se propone la realización de tareas opcionales, que serán tenidas en cuenta en la evaluación, pero no son necesarias para obtener el aprobado en la práctica.

- **Exportación de billetes a formato PDF:** esta opción permite generar un fichero PDF con los datos de un billete comprado por el usuario, de manera que éste pueda guardarlo. Para ello será necesario utilizar alguna librería de generación dinámica de documentos, por ejemplo [Apache FOP](#).
- **Funcionalidad de autocompletar con Ajax:** esta función sugerirá de forma automática, a medida que el usuario va tecleando el nombre de una estación, nombres de estación que coincidan con las letras introducidas hasta el momento. Si se trata del campo de estación destino, sólo se sugieren aquellas que tengan una línea con la estación de origen dada.
- **Otras mejoras con AJAX:** cualquier mejora a la interfaz de la aplicación que realice peticiones `XmlHttpRequest` al servidor y modifique el documento dinámicamente mediante la API DOM de JavaScript.
- **Fechas de viaje flexibles:** cuando no haya ningún servicio disponible para el trayecto especificado por el usuario, se mostrarán botones que permitan consultar si existen billetes un día antes o un día después.
- **Uso de transacciones en la base de datos:** para garantizar que en ningún caso se vendan más billetes que plazas disponibles hay, en el instante en que el sistema va a emitir un billete debería realizar de forma atómica la comprobación de plazas en el vagón seleccionado, su decremento y la introducción del billete en la tabla `Billetes`. Se pide implementar un sistema robusto de emisión de billetes mediante el soporte a transacciones de InnoDB. Puede obtenerse más información en [el manual de referencia de MySQL](#).
- **Pool de conexiones:** dado que una aplicación Web suele ser capaz de responder peticiones de usuarios de forma concurrente, puede necesitar realizar consultas a la base de datos concurrentemente. Una forma muy eficiente de gestionar estas conexiones es utilizar lo que se conoce como *pool* de conexiones, que no es más que un gestor que maneja un grupo de conexiones JDBC reutilizables. Tomcat proporciona una implementación de *pool* de conexiones, por lo que su uso requiere únicamente configurar adecuadamente la aplicación Web (y recordar cerrar en el código de la aplicación las conexiones cuando dejen de ser necesarias, para que el *pool* las recupere y permita reutilizarlas). Puede obtenerse más información en la [documentación de Tomcat](#).
- **Integración con Google Maps u otro servicio de información geográfica similar:** mostrar en la interfaz un mapa posicionando las distintas estaciones en que la compañía presta servicio. Para ello se puede utilizar, por ejemplo, [Google Maps API](#). Para realizar esta mejora, se permite crear una nueva tabla en el modelo de datos que relacione estaciones con sus coordenadas de longitud y latitud.
- **Integración con Flickr u otro servicio de almacenamiento de fotografías:** una vez comprado un billete, mostrar en la interfaz fotografías de la ciudad de destino, obtenidas en tiempo real desde Flickr a través de su API, o de un servicio de fotografías similar que también proporcione API. Flickr proporciona una API de búsqueda a la que se puede dar como parámetro un identificador de lugar, y devuelve fotografías de dicho lugar (el identificador de lugar se puede obtener a partir del nombre de ciudad y país mediante otra operación de la API). La invocación a la API de Flickr se puede hacer bien desde el servidor con código Java, bien desde el cliente con código JavaScript y JSON. Más información acerca de la [API de Flickr](#).

El objetivo de estas funciones adicionales es que el alumno explote su capacidad para resolver los problemas que se le planteen de forma autónoma. Por ello, la ayuda de los profesores con respecto a estas funciones será más limitada que con respecto a las funciones mínimas. La calificación de las funciones adicionales depende de la dificultad asociada a su implementación, por lo que no todas se valorarán con la misma puntuación. Se proporciona una lista grande de funciones opcionales para que haya más variedad de opciones. Obviamente, no es necesario implementar todas para obtener la máxima calificación.

Los alumnos pueden proponer funciones adicionales que no se detallan en este enunciado, pero es necesario consultar a los profesores acerca de la posible valoración de las mismas antes de implementarlas.

6. Recomendaciones de diseño

Se aconseja seguir las siguientes recomendaciones para facilitar el desarrollo de la práctica:

1. Diseñar la aplicación Web de tal forma que reutilice la mayor cantidad posible de código de la aplicación Swing.
2. Separar las tareas de procesamiento de peticiones y obtención de resultados de las tareas de presentación de resultados. Las primeras debes realizarlas mediante servlets. Las de presentación, mediante documentos JSP. Como norma general, los servlets no deberían generar código XHTML, mientras que los documentos JSP deberían tener la menor cantidad de código Java posible. La forma de trabajo recomendada, para tareas que impliquen procesamiento (por ejemplo, interacción con la base de datos) es la siguiente:
 - Un servlet recibe la petición del usuario, la procesa y genera los resultados.
 - El servlet pasa el control a una página JSP mediante el mecanismo de `forward`. Le puede pasar la información a mostrar mediante beans guardados en el ámbito (*scope*) que resulte más adecuado. Por ejemplo, los datos que sean relevantes sólo durante el procesamiento de la petición actual se pasarían como atributos del objeto `HttpServletRequest` (ámbito `request`).
 - La página JSP recupera estos beans y genera la presentación de los resultados.
3. Programar un servlet por cada operación básica, en vez de un servlet que se encargue de varias operaciones distintas.
4. Escribir trazas mediante el método `log` de las clases `ServletContext` o `GenericServlet` para depurar la aplicación. Las trazas serán accesibles en el directorio `logs` de la instalación de Tomcat. También se puede depurar la aplicación (puntos de ruptura, ejecución paso a paso, visualización del contenido de las variables, etc.) mediante Eclipse (con plug-in WTP, ya instalado en los laboratorios) y otros entornos de desarrollo avanzados.
5. Utilizar siempre URLs relativas para que la aplicación sea totalmente portable. Para facilitar esto, es recomendable acceder a los JSPs y servlets al mismo nivel de URL. Esto puede hacerse declarando el `servlet-mapping` de los servlets en `WEB-INF/web.xml` como (es un ejemplo) `"/CrearIncidencia"`, pero no como `"/servlet/CrearIncidencia"`.

7. Formato de entrega

La práctica se entregará a través del sistema Web que se habilitará al efecto, cuya URL se publicará en la página principal de la Web de la asignatura.

Se entregará un único fichero con extensión `.war` (comprimido con `JAR`), que contenga la aplicación web lista para ser desplegada (con todos los ficheros de configuración, clases compiladas, imágenes, hojas de estilo, etc.) Adicionalmente, en el directorio `WEB-INF/src` del fichero `WAR` se incluirá el código fuente de todas las clases Java utilizadas.

El fichero `WAR` contiene toda la estructura de la aplicación Web comprimida desde dentro del directorio principal de la misma:

```
cd ${CATALINA_HOME}/webapps/miaplicacion/  
jar cvf miaplicacion.war *
```

Para reducir el tamaño de la entrega, este fichero no debe contener datos innecesarios como los siguientes:

- El fichero `JAR` del conector de `JDBC`, si se usa el recomendado en este enunciado.

- Ficheros de copia de seguridad generados por algunos editores (por ejemplo, *.bak, *~, #*, etc.).

Independientemente de que se haya desarrollado la aplicación en un entorno integrado como Eclipse o NetBeans, los autores de la práctica deben ser capaces, el día de la entrevista de evaluación, de compilar su aplicación desde línea de comandos. Aquellos que no sepan hacerlo pueden preguntar a los profesores en las prácticas o en tutorías, antes de vencer el plazo de entrega.

8. Criterios de evaluación

La calificación obtenida en esta práctica estará entre 0 y 10, y supondrá un 25% de la calificación final en la asignatura. Es necesario aprobar esta práctica (obtener al menos un 5) para aprobar la asignatura.

Parte de la evaluación se basará en una entrevista personal de los profesores con los autores de la práctica, que se realizará con cita previa en los horarios de clases prácticas. Los dos miembros de la pareja de prácticas deben demostrar conocimientos suficientes de la práctica que hayan entregado.

Además de que se implemente correctamente la funcionalidad obligatoria, se tendrán en cuenta los siguientes aspectos en la evaluación:

- Funcionalidad opcional implementada, tanto la propuesta en este enunciado como la desarrollada a iniciativa de los alumnos.
- Calidad y elegancia del diseño y código de la aplicación. Uso adecuado de la orientación a objetos. Gestión de errores adecuada.
- Usabilidad de la aplicación.

Para obtener una calificación de 5 o superior es necesario implementar la funcionalidad mínima expuesta en este enunciado. Por otra parte, un programa que implemente sólo la funcionalidad mínima puede optar hasta una calificación máxima de entorno a 8 teniendo en cuenta el resto de aspectos a valorar. El resto de la calificación, hasta 10, puede obtenerse implementando funcionalidad opcional.

9. Referencias y material complementario

9.1. Estándares

- [API Servlet 2.5](#)
- [RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1](#)
- [XHTML1.0: The Extensible HyperText Markup Language](#)
- [XHTML 1.1 - Module-based XHTML](#)
- [CSS 2.1 Specification](#)

9.2. Servlets y JSP

- [Apache Tomcat](#)
 - [Apache Tomcat 6.0: Documentación](#)
 - [Servlets en java.sun.com](#)
 - [Servlet Essentials](#)
 - [Story of a Servlet: A tutorial](#)
 - [Manual de Servlets y JSPs](#)
-

- [Java Server Pages en java.sun.com](#)
- [Manuales de Servlets, JSP y JDBC \(nivel básico/intermedio, nivel avanzado\)](#).
- [Manual de JSP](#)

Tomcat incluye ejemplos de Servlets y JSPs encapsulados en la aplicación Web `examples`. Están accesibles en el directorio `webapps/examples` de la instalación de Tomcat, y a través de un navegador Web si Tomcat está en ejecución:

```
http://localhost:8080/examples/
```

9.3. MySQL

- [Manual de Referencia de MySQL](#)
- [MySQL Connector/J 5.1](#)
- [JDBC Database Access](#)