Práctica 1 DAI: Aplicaciones Web con ASP, JSP, Ajax y Java Applets

Introducción

En esta práctica vamos a desarrollar una **aplicación web para una agencia de viajes** que permita la realización de una serie de gestiones relacionadas con la **reserva de viajes on-line**. La web dispondrá de una **zona accesible a cualquier usuario**, con información de la empresa, ubicación, imagen corporativa, búsqueda de vuelos que cumplen una condición, etc. y **otra zona, accesible únicamente tras llevar a cabo un proceso de identificación**, que permitirá a un usuario consultar las reservas realizadas.

Para el desarrollo de la parte "estática" del sitio emplearemos **XHTML** y **CSS**, mientras que para la parte dinámica emplearemos **Javascript** y la tecnología **ASP** con lenguaje **VBScript**.

La aplicación empleará una sencilla **base de datos**, desarrollada en **Firebird** cuya plantilla básica está disponible en la carpeta drive y a la que progresivamente le iremos añadiendo funcionalidad. El acceso deberá realizarse a través de **ODBC**. Todos los drivers y ejecutables necesarios se encuentran disponibles en la web de la asignatura (http://umh2806.edu.umh.es).
Dispondrá únicamente de las siguientes tablas:

- **Ciudad**: con información de las diferentes ciudades desde las que, y hacia las que, se pueden realizar vuelos. Debe almacenar al menos el precio asociado a las tasas de aeropuerto de dicha ciudad.
- **Compañía**: información de las compañías aéreas con las que opera la empresa.
- **Vuelo**: contiene toda la información relacionada con los vuelos: Ciudad origen, Destino, Fecha y hora de salida, Compañía, duración del vuelo, Nº plazas disponibles, Avión,…
- **Avión**: Tabla en la que se registran los diferentes modelos de avión que incluirá como mínimo el número de plazas del mismo y el precio base del vuelo.
- **Reserva**: Contiene un registro de las solicitudes de reserva de vuelo que se realizan. Cada nuevo registro deberá generar un número identificativo único del número de reserva, Nombre y Apellidos de la persona, DNI, Vuelo, Nº de Asientos reservados…

Parte obligatoria

Desarrollo de la interfaz básica (4 puntos)

El sitio web de la agencia de viajes debe permitir:

GESTIÓN DE CIUDADES:

Para acceder a este apartado será necesario realizar previamente un **acceso identificado de administrador**. Una vez logueado correctamente, esta opción mostrará un **listado de todas las ciudades de la base de datos** desde las que, y hacia las que, se pueden realizar vuelos.

Al hacer clic sobre una de las ciudades de la lista, se accederá a la ficha **Detalles de la Ciudad**, donde podrán **modificarse** todos los datos de la ciudad, excepto el **IDCIUDAD** que **será asignado de forma automática**.

Además se mostrará un botón que permita **Añadir una nueva ciudad**.

También se incluirá en este apartado un botón que permita **Eliminar Ciudades** de la base de datos.

GESTIÓN DE RESERVAS

Sin necesidad de identificarse, cualquier usuario podrá **consultar vuelos disponibles** mediante un formulario de consulta. **Ningún dato es obligatorio**, por lo que si se dejan en blanco, la aplicación deberá devolver una lista con todos los vuelos disponibles; o bien un listado de vuelos que cumpla las condiciones

expresadas, si se rellenan los campos del formulario. Se debe mostrar el precio del vuelo según se a continuación: **Precio = (Nº Plazas del avión – Nº Plazas disponibles)/nº Plazas del avión * Precio Base del vuelo + Tasas aeropuerto**. De este modo el precio se incrementa a medida que disminuye el número de plazas disponibles.

Una vez obtenido el listado de vuelos disponibles, el usuario podrá **realizar una reserva sobre alguno de los vuelos localizados**:

- Se deberá comprobar que quedan plazas suficientes y actualizar el número de plazas disponibles tras realizar la reserva.
- **Se devolverá al usuario un identificador único de la reserva** realizada para posteriores consultas y **se le indicará el precio tras seleccionar el número de asientos** que desea reservar.

Tras la reserva del vuelo de ida, la interfaz de consulta de vuelos **deberá mostrar también los posibles vuelos de vuelta**, permitiendo por tanto al usuario realizar una segunda reserva con el vuelo de vuelta.

Además, **previo acceso identificado de cliente**, el usuario podrá consultar reservas realizadas **a partir del DNI y del número de reserva generado en el paso anterior**. Una vez mostrados los datos de la reserva **se debe poder cancelar**, en cuyo caso el número de plazas disponibles en el vuelo se debe incrementar. Si faltan menos de 48 horas para la salida del vuelo, la aplicación mostrará un mensaje de error indicando que ha finalizado el plazo máximo de cancelación.

Inclusión de Técnicas Ajax en la interfaz de usuario (2 puntos)

Una vez finalizada la versión básica de la interfaz, tal y como se pide en el apartado anterior, se desarrollará una versión alternativa de la misma, **que permita la búsqueda de vuelos empleando técnicas AJAX**. De manera que:

- **Al hacer clic en una de las ciudades para ser editada**, el formulario de edición de los datos de dicha ciudad **deberá cargarse en segundo plano**, sin necesidad de recargar la página.
- La **recarga de datos** en el cuadro de lista desplegable de **ciudad de destino**, se realizará en segundo plano, **sin recargar la página**, al cambiar la ciudad de origen, mostrando sólo aquellas ciudades para las que existan vuelos.
- Del mismo modo, **la tabla que contiene los resultados de la búsqueda de vuelos** deberá rellenarse en segundo plano **sin necesidad de recargar la totalidad de la página**.

Parte opcional

Desarrollo de la interfaz Java para el administrador (2 puntos)

En esta parte de la práctica vamos a desarrollar la interfaz que **permitirá a los usuarios con perfil de administrador añadir, consultar y modificar información** de la base de datos. Para ello se diseñará, empleando **Java** y el **AWT**, un **Applet**, con un **interfaz gráfico que se ejecutará en el lado del cliente** y una **página ASP**, que recibirá la información desde el Applet y se encargará de **ejecutar las sentencias SQL** sobre la base de datos, empleando objetos **ADO**.

Se debe añadir al sitio desarrollado en el apartado anterior las siguientes páginas y clases de Java:

- **ServApplet.asp**: Página ASP que recibirá, en forma de peticiones **GET**, información del Applet y que devolverá a éste la información solicitada (si es una consulta) o se encargará de insertar o actualizar los registros (si se trata de una inserción o modificación). Los **parámetros** serán:
  - **Accion**: con tres posibles valores: **Consulta**, **Insercion**, **Modificacion**.
  - **NombreCampo** = Valor: Lista de campos con sus respectivos valores para insertar en la tabla de vuelos.
    - Cada parámetro se separará de los demás empleando el carácter "**&**", según la especificación del protocolo HTTP.
- **Administrador.asp**: página a la que será redirigido el usuario con perfil de administrador, cuando se valide como tal en la base de datos, y que únicamente se empleará para **contener el Applet** especificado a continuación.
- **CliApplet.class**: Applet de java **con interfaz gráfico** para permitir al administrador insertar, consultar o modificar vuelos. Empleando botones de comandos, menús, controles de edición de texto, cuadros de lista, checkbox, o cualquier otro componente disponible, debe permitir al administrador:
  - **Insertar un nuevo vuelo**
  - **Visualizar un vuelo a partir** del identificador del mismo
  - **Modificar cualquiera de los campos de un vuelo** (salvo el identificador de vuelo)

Consulta de reservas realizadas en JSP (2 puntos)

En esta parte de la práctica vamos a **desarrollar una nueva funcionalidad** empleando tecnología **JSP y Técnicas AJAX**. Dado que ya tenemos un IIS escuchando en el puerto 80, será necesario configurar el servidor **Tomcat** en un puerto distinto, por ejemplo, **8081**.

Se añadirá una nueva página, accesible desde la página principal, **a través de un acceso identificado de cliente**, que muestre un listado de reservas realizadas a partir del **DNI** de la persona que la realizó y de una **ciudad de origen**. Será necesario implementar un formulario que solicite el DNI y la ciudad de origen del vuelo, y además muestre un botón que será el encargado de **llamar a la página JSP encargada de recuperar los datos**.

Los datos **se mostrarán en una tabla que incluirá**, al menos, columnas para IdReserva, IdVuelo, Apellidos, Nombre, Ciudad Origen, Ciudad Destino, Fecha del Vuelo, Compañía y Estado (Cancelado o No Cancelado); se puede añadir cualquier otro dato que consideréis interesante.

La tabla con las reservas resultante **deberá ser cargada en segundo plano** mediante técnicas AJAX, **sin que sea recargada la página**.

Normas de entrega

La entrega de la práctica se hará en un **único fichero .ZIP**, que incluirá todas las páginas implementadas, ficheros java, jsp, asp, imágenes y memoria descriptiva del sitio desarrollado. Para ello se empleará la herramienta de tareas de la web de la Universidad.

Apéndices

Carga dinámica de un Cuadro de lista

Para la implementación del formulario de búsqueda de vuelos habrá que emplear varios controles de lista desplegable. Deberán obtener sus datos de forma dinámica a partir de la información contenida en las correspondientes tablas. A continuación tenéis un ejemplo de cómo hacerlo:

```
…
<SELECT NAME="Cuadro_Lista">
<%
   'Recorreremos el RecordSet
   do while not rsTablaDatos.EOF
      Response.Write("<OPTION VALUE='" + rsTablaDatos("CampoDatos") + "'>")
      Response.Write(rsTablaDatos("CampoDatos") + "</OPTION>")
      'Nos desplazamos por el RecordSet
      rsTablaDatos.MoveNext
   loop
%>
</SELECT>
```

Ejemplo comunicación applet

En el siguiente código tenéis un ejemplo de las clases de java necesarias para realizar la comunicación entre el applet y la página asp y establecer un flujo de datos. Se solicita a la página **ServApplet .asp** toda la información del vuelo con **IdVuelo** = 1 (el nombre del campo dependerá del diseño de la B. de D.). La página ASP devolvería cada uno de los campos de la consulta en una línea diferente, aunque podéis implementarlo de cualquier otra forma que se os ocurra: Empleando un carácter separador, etc…

La comunicación se realiza empleando GET (aunque, con ligeras modificaciones, también podría hacerse con POST), por lo que desde la página asp se debería acceder a los parámetros de la petición empleando el método **QueryString** del objeto **Request.**

En el ejemplo, la información recibida se muestra en un **TextArea**, en la práctica, deberán rellenarse adecuadamente los controles empleados en la interfaz.

```
import java.net.*;
import java.io.*;
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class CliApplet extends Applet{
   public void init(){
      String hostName, applicationPath, webServerStr, applicationGet;
      int port;
      TextArea ta = new TextArea( 20, 40 );
      add(ta);
      applicationPath = "ServApplet.asp";
      try {
         URL hostURL = getCodeBase();
         hostName = hostURL.getHost();
         port = hostURL.getPort();
         if (port == -1)
            port = 80;
         webServerStr = "http://" + hostName + ":" + port + applicationPath;
         applicationGet = webServerStr + "?" + URLEncoder.encode("Accion") + "=" +
         URLEncoder.encode("Consulta") + "&" +
         URLEncoder.encode("IdVuelo") + "=" +
         URLEncoder.encode("1") ;
         URL servApplet = new URL(applicationGet);
         URLConnection servAppletConnection = servApplet.openConnection();
         DataInputStream dis = new DataInputStream(servAppletConnection.getInputStream());
         String inputLine;
         while ((inputLine = dis.readLine()) != null) {
            ta.append(inputLine + "\n");
         }
         dis.close();
      }catch (MalformedURLException me) {
         System.out.println("MalformedURLException: " + me);
```

```
      }catch (IOException ioe) {
         System.out.println("IOException: " + ioe);
      }
   }
}
```

Recuperación de datos JSP

Para recuperar los datos requeridos y mostrarlos al usuario podéis emplear un procedimiento almacenado que muestra todas las reservas. Desde la página JSP emplearíais un clausula where para filtrar únicamente las del DNI especificado.

Como modelo podéis emplear el archivo pruebaConexion.jsp que tenéis en la carpeta drive y que realiza algo parecido sobre la tabla ciudades a partir del nombre de la ciudad.

Éste es el código del procedimiento que podéis usar. Podéis cambiar lo que consideréis oportuno si queréis cambiar los datos a mostrar:

```
CREATE PROCEDURE LISTADO_RESERVAS
RETURNS (
    IDRESERVA INTEGER,
    FECHA_VUELO DATE,
    CIUDAD_ORIGEN VARCHAR (50) CHARACTER SET WIN1251,
    CIUDAD_DESTINO VARCHAR (50) CHARACTER SET WIN1251,
    CANCELADO SMALLINT,
    COMPANIA VARCHAR (50) CHARACTER SET WIN1251,
    NIF VARCHAR (12) CHARACTER SET WIN1251)
AS
declare variable IdVuelo integer;
declare variable IdCiudadOrigen integer;
declare variable IdCiudadDEstino integer;
declare variable IdCompania integer;
BEGIN
  for select IDRESERVA, CANCELADO, IDVUELO, NIF
  from reserva
  into :IdReserva, :Cancelado, :IdVuelo, :Nif
  do begin
     select IDCIUDADORIGEN, IDCIUDADDESTINO, FECHA, IDCOMPANIA
     from vuelo
     where IDVUELO = :IdVuelo
     into :IdCiudadOrigen, :IdCiudadDestino, :Fecha_Vuelo, :IdCompania;

     select CIUDAD
     from ciudad
     where IDCIUDAD = :IdCiudadOrigen
     into :Ciudad_Origen;

     select CIUDAD
     from ciudad
     where IDCIUDAD = :IdCiudadDestino
     into :Ciudad_Destino;

     select COMPANIA
     from compania
     where IDCOMPANIA = :IdCompania
     into :Compania;

     SUSPEND;
  end
END
```