

SISTEMAS DE INFORMACIÓN

PRÁCTICA 3

Diseño, desarrollo e instalación de un sistema de información Web

Álvaro Juan Ciriaco (682531)
Alejandro Guiu Pérez (680669)

ÍNDICE:

1. INTRODUCCIÓN Y OBJETIVOS	2
2. DISEÑOS REALIZADOS	2
I. INDEX.....	2
II. INICIAR SESIÓN	3
III. REGISTRARSE.....	3
IV. CONSULTAR.....	4
V. COMPARAR.....	5
VI. PERFIL.....	5
VII. ACTIVIDADES.....	6
3. PROGRAMACIÓN DE LOS SERVLETS.....	7
I. LOGIN-USER.....	7
II. INSERT-USER.....	7
III. LOG-OUT.....	8
IV. ACTIVIDADES-ORDENADAS.....	8
V. ACTUALIZAR-USUARIO.....	8
VI. COMPLETAR-PERFIL.....	9
VII. COMPARAR.....	10
VIII. CONSULTAR.....	11
IX. ELIMINAR-USUARIO.....	11
X. RESERVAR.....	12
4. METODOLOGÍA DE TRABAJO.....	12
I. RECURSOS.....	12
II. HERRAMIENTAS UTILIZADAS.....	12
III. DISTRIBUCIÓN DEL TRABAJO.....	12
5. DIFICULTADES ENCONTRADAS.....	13
6. MANUAL DE INSTALACIÓN.....	13
7. MANUAL DE USUARIO.....	14

1. Introducción y objetivos:

En esta tercera práctica se han trabajado los conceptos básicos de programación Web con persistencia de datos. Para ello se ha utilizado un sistema gestor de bases de datos relacionales compatible con JDBC (en este caso Oracle). Además se ha completado el desarrollo del sistema de información realizado en las sesiones de prácticas anteriores (Práctica 1 y Práctica 2).

Entre los principales objetivos de la práctica se encuentran:

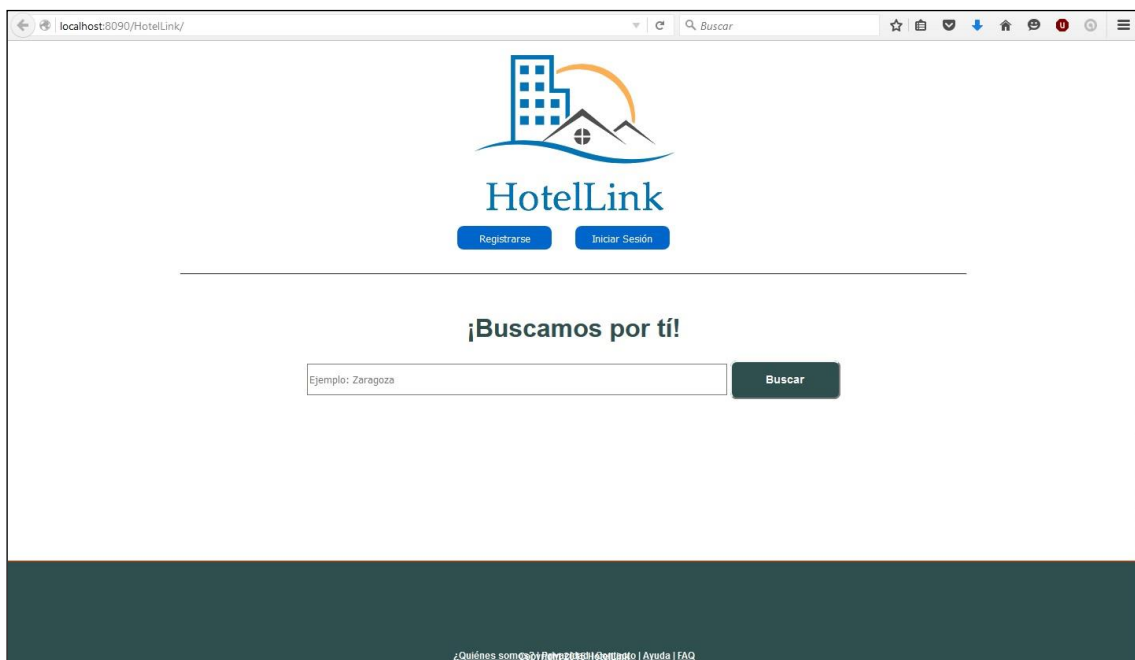
- Configurar el servidor web y la conexión con el sistema de gestión de bases de datos (Oracle).
- Realizar el diseño de la capa de persistencia de datos, en la que se agrupan las clases que gestionan la interacción con el sistema de almacenamiento desarrollado en la práctica 2.
- Diseñar la capa modelo de la aplicación web cuya interfaz se ha diseñado en la práctica 1.

El entorno empleado en el desarrollo de la práctica es JDK 1.7, Tomcat y Oracle como gestor de bases de datos relacionales.

2. DISEÑOS REALIZADOS:

2.1. INDEX:

Archivo JSP que contiene el diseño y funcionalidad de la página principal de la web. En ella se tiene la posibilidad de iniciar sesión con un nombre de usuario y contraseña (que se encuentren en la DB), registrarse como nuevo usuario para así poder realizar reservas de hoteles, o la opción de buscar sin iniciar sesión hoteles que se encuentren

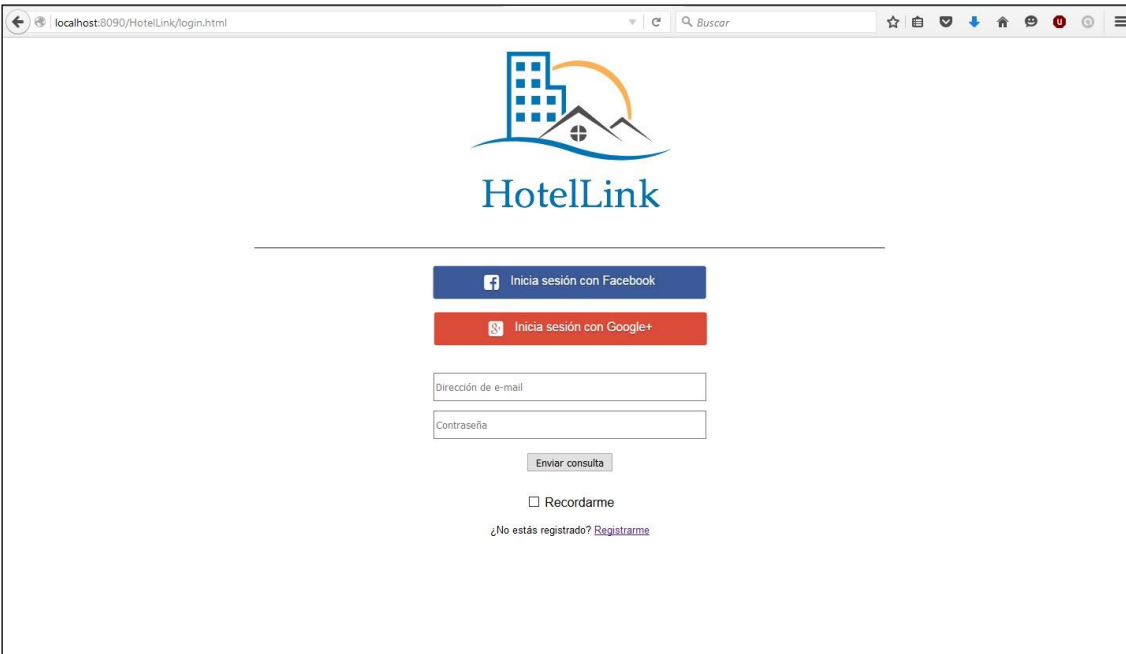


en una población específica (las características y fechas de estos hoteles se podrán incluir en la búsqueda posteriormente).

2.2. INICIAR SESIÓN:

Archivo JSP que contiene el diseño y funcionalidad de la página encargada de gestionar el acceso a una cuenta de usuario incluida con anterioridad en la base de datos del servidor. En ella se le permite al cliente introducir un nombre de usuario (email) y una contraseña válida. En caso de que los datos sean correctos, se le informa al cliente de que ha iniciado sesión de forma correcta y se le reedirecciona a la página principal. En caso contrario, se le muestra por pantalla que los datos introducidos no son correctos.

El usuario también dispone de la posibilidad de iniciar sesión mediante una red social, como Facebook o Google+.



localhost:8090/HotelLink/login.html

HotelLink

Inicia sesión con Facebook

Inicia sesión con Google+

Dirección de e-mail

Contraseña

Enviar consulta

☐ Recordarme

¿No estás registrado? [Regístrate](#)

2.3. REGISTRARSE:

Archivo JSP que contiene el diseño y funcionalidad de la página encargada de gestionar el registro de nuevos usuarios en la página web. En ella se le permite al usuario introducir todos los datos necesarios para la creación de una nueva cuenta de usuario. Este debe introducir un nombre y apellidos válidos, una contraseña, un correo electrónico con formato válido y una dirección. El resto de datos no son necesarios para el registro, aunque pueden ser añadidos por comodidad.

Una vez introducidos los datos el servidor se encarga de añadir el nuevo usuario a la BD. Si ya existe un usuario con el mismo correo electrónico se le informa del error al usuario. En el caso de que el correo electrónico introducido no sea válido o la contraseña no coincida con el valor de "Repetir contraseña" introducido en el formulario se le muestra al usuario un error, indicando que dato ha introducido de forma errónea.

2.4. CONSULTAR:

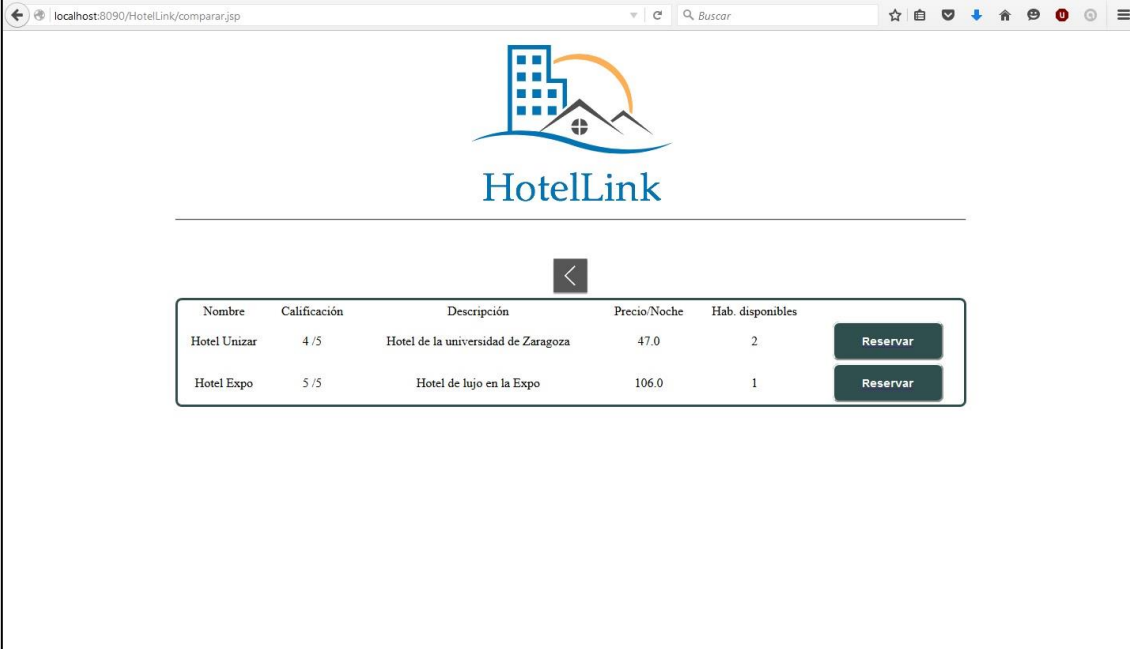
Archivo JSP que contiene el diseño y funcionalidad de la página encargada de mostrar el resultado de la búsqueda realizada por el usuario. En ella se da la opción de añadir nuevos filtros para la búsqueda, entre los que se encuentran las fechas exactas de inicio y fin de la reserva del hotel, y el número de personas. Según estos nuevos valores, se mostrará por pantalla una lista con los hoteles que tengan habitaciones libres (con tamaño suficiente para las personas introducidas) entre las fechas indicadas.

Además, el usuario tiene la posibilidad de ordenar la lista de hoteles por precio de menor a mayor, facilitando así la búsqueda.

2.5. COMPARAR:

Archivo JSP que contiene el diseño y funcionalidad de la página encargada de mostrar la comparación entre los hoteles seleccionados por el usuario.

En esta comparación se muestra de forma clara el precio del hotel, las estrellas, la localización, las habitaciones disponibles, el nombre y una breve descripción de este.



Nombre	Calificación	Descripción	Precio/Noche	Hab. disponibles	
Hotel Unizar	4 / 5	Hotel de la universidad de Zaragoza	47.0	2	<button>Reservar</button>
Hotel Expo	5 / 5	Hotel de lujo en la Expo	106.0	1	<button>Reservar</button>

2.6. PERFIL:

Archivo JSP que contiene el diseño y funcionalidad de la página encargada de mostrar el perfil del usuario que se encuentra logeado actualmente.

En ella se muestran los datos que la BD tiene sobre el usuario, entre los que se encuentran: nombre y apellidos del usuario, correo electrónico, contraseña, localidad, sexo, código postal, y dirección.

Si el usuario desea cambiar alguno de estos datos, ya sea porque son erróneos o simplemente desea cambiarlos, únicamente tiene que modificarlos en el perfil y aceptar los cambios. Esto automáticamente hará que los datos de la base de datos sean actualizados.

localhost:8090/HotelLink/perfil.jsp

HotelLink

Perfil Mis Actividades Cerrar Sesión Borrar Cuenta

Nombre*: Javier
 Apellidos*: JavierApellidos
 Sexo: ☒ Hombre ☐ Mujer
 Correo electrónico*: javier@correo.com
 Repetir correo electrónico*:
 Contraseña*:
 Repetir contraseña*:
 Dirección*:
 Localidad*: Zaragoza
 Código postal*:
 Nota: Los campos marcados con un asterisco deben rellenarse de forma obligatoria.
 Enviar consulta

Si la contraseña se modifica y esta no coincide con el valor de “Repetir contraseña”, o el nuevo correo electrónico no es válido o está en uso, no se realizarán los cambios.

2.7. ACTIVIDADES:

Archivo JSP que contiene el diseño y funcionalidad de la página encargada de mostrar las reservas realizadas por un usuario.

localhost:8090/HotelLink/ActividadesOrdenadas

HotelLink

Fecha inicio: 01 01 2015 Fecha final: 01 01 2015 Buscar

Perfil Mis Actividades

Hotel Expo
 Hotel de lujo en la Expo
 Hotel Unizar
 Hotel de la universidad de Zaragoza

Se da la opción al usuario de buscar las reservas que se encuentren entre una fecha de inicio y otra fecha final, facilitando así la búsqueda.

3. PROGRAMACIÓN DE LOS SERVLETS:

Una de las tareas más complicadas es la programación de los Servlets de la página web. Los servlets utilizados son los siguientes:

3.1. LOGIN-USER:

Servlet encargado del inicio de sesión de un usuario registrado con anterioridad en la base de datos de la web. Este ejecuta el método consultarUsuario que se encarga de realizar una sentencia SELECT en la base de datos. Si existe un usuario con el correo electrónico y contraseña insertados por el usuario en el proceso de login se devuelve TRUE, en caso contrario FALSE (mostrando así que los datos introducidos por el usuario no son los correctos).

```
public boolean consultarUsuario(String correo, String password) {  
    String sentenciaSQL = "SELECT COUNT(*) FROM USUARIO WHERE EMAIL='" +  
        correo + "' AND PASSWORD='" + password + "'";  
    int resultado = Integer.parseInt(ot.executeQueryReturn(sentenciaSQL).get(0));  
    if (resultado==1) return true;  
    else return false;  
}
```

Una vez realizada la consulta, si los datos introducidos por el usuario son los correctos se crea un atributo de sesión en el cual se almacena el correo del usuario, indicando así que un usuario está logeado actualmente. Esto hace que aparezcan opciones como la de visitar el perfil, desconectarse, borrar la cuenta, ver el historial de reservas, reservar nuevos hoteles...

3.2. INSERT-USER:

Servlet encargado de registrar un nuevo usuario en la base de datos de la página web. Este ejecuta el método insertarUsuario que se encarga, en primer lugar de realizar una sentencia SELECT en la base de datos comprobando si ya existe un usuario con ese correo electrónico, si existe cancela el registro, mientras que si no existe realiza una sentencia de tipo INSERT, la cual inserta en la base de datos un nuevo usuario con los datos introducidos por el cliente en el formulario de registro.

```
public boolean insertarUsuario(String nombre, String apellidos, String sexo,  
    String correo, String password, String localidad, String postal,  
    String direccion) {  
    if(Integer.parseInt((ot.executeQueryReturn("SELECT COUNT(*) FROM USUARIO WHERE EMAIL='"+correo+"'").get(0)))==0){  
        ot.executeSentence("INSERT INTO USUARIO(EMAIL,NOMBRE,APELLIDOS,SEXO,PASSWORD,"  
            + "PAIS,LOCALIDAD,DIRECCION,CODIGO_POSTAL) VALUES (?,?,?,?,?,?,?,?,?)",  
            correo, nombre, apellidos, sexo, password, localidad, localidad,  
            direccion, postal);  
        return true;  
    } return false;  
}
```

Si los datos introducidos por el usuario en el formulario no son correctos, no se ejecuta la función insertUsuario y se le proporciona al usuario el motivo por el cual no se ha podido registrar. Sin embargo, si el usuario consigue registrarse de forma correcta,

también se le notifica con la finalidad de que el usuario se mantenga informado constantemente de cualquier cambio.

3.3. LOG-OUT:

Servlet encargado de cerrar la sesión actual de un usuario en la página web. Este se encarga de borrar los atributos asignados a la sesión actual, eliminando así cualquier rastro del usuario que estaba navegando por la web.

Esto hace que ya no aparezcan las opciones de cerrar sesión, borrar cuenta, listar actividades del usuario, reservar hoteles... sin embargo permite volver a registrarse o iniciar sesión de nuevo.

3.4. ACTIVIDADES-ORDENADAS:

Servlet encargado de mostrar al usuario una lista de las últimas reservas realizadas, pudiendo mostrarse las reservas realizadas durante un periodo exacto de tiempo.

```
public ArrayList<ResultadoBusqueda> listaActividadesOrdenadas (String usuario, String diaEnt, String mesEnt, String anoEnt,
String diaEnt2, String mesEnt2, String anoEnt2){

    String sentenciaSQL = "SELECT hotel.nombre FROM actividad,usuario,hotel WHERE email='javier@correo.com' AND " +
    "actividad.usuario=usuario.email AND hotel.id=actividad.hotel_id AND " +
    "((((" + anoEnt + "<actividad.fecha2) AND (actividad.fecha2<" + anoEnt2 + ")) OR (((" + anoEnt + "=actividad.fecha2) AND " +
    "(" + anoEnt2 + "!actividad.fecha2) AND (" + mesEnt + "<=actividad.fecha1)) OR (((" + anoEnt + "=fecha2) AND " +
    "(" + anoEnt2 + "=fecha2) AND (" + mesEnt + "<=fecha1) AND (fecha1<=" + mesEnt2 + ")) OR (((" + anoEnt2 + "=fecha2) AND " +
    "(fecha2!=" + anoEnt + ") AND (fecha1<" + mesEnt2 + "))))";

    ArrayList<ResultadoBusqueda> resultados = new ArrayList<ResultadoBusqueda>();
    ArrayList<String> resultadosString = ot.executeQueryReturn(sentenciaSQL);
    int numResultados = resultadosString.size();
    while(numResultados>0){
        sentenciaSQL = "SELECT hotel.nombre, estrellas, descripcion, tipo_habitacion FROM hotel, actividad WHERE " +
        "hotel.nombre=" + resultadosString.get(numResultados-1) + " AND hotel.id=actividad.hotel_id";
        ArrayList<String> res = ot.executeQueryReturn(sentenciaSQL);
        resultados.add(new ResultadoBusqueda(res.get(0),Integer.parseInt(res.get(1)),res.get(2),null,
        null, res.get(3),-1,-1,-1));
        numResultados--;
    }
    return resultados;
}
```

Este ejecuta el método listaActividadesOrdenadas el cual se encarga de realizar una consulta de tipo SELECT a la base de datos que devuelve los hoteles reservados por el usuario actual entre la fecha de inicio y final indicada.

3.5. ACTUALIZAR-USUARIO:

Servlet encargado de actualizar la información de perfil del usuario logeado en la página web actualmente.

Este se encarga de llamar al método actualizarUsuario que recibe como parámetros los nuevos valores a introducir en la base de datos. Primero se encarga de comprobar si estos son distintos de null (si son iguales no modifica los valores actuales de la base de datos), si esto se cumple actualiza el valor y lo guarda de forma permanente hasta que el usuario desee modificarlo de nuevo o se elimine la cuenta de la base de datos.

```

public void actualizarUsuario(String nombre, String apellidos, String sexo,
String correo, String password, String localidad, String postal,
String direccion) {
    if (!nombre.equals(null)) {
        String sentenciaSQL = "UPDATE USUARIO SET NOMBRE='" +
nombre + "' WHERE EMAIL='" + correo + "'";
        ot.executeSentence(sentenciaSQL);
    }
    if (!apellidos.equals(null)) {
        String sentenciaSQL = "UPDATE USUARIO SET APELLIDOS='" +
apellidos + "' WHERE EMAIL='" + correo + "'";
        ot.executeSentence(sentenciaSQL);
    }
    if (!sexo.equals(null)) {
        String sentenciaSQL = "UPDATE USUARIO SET SEXO='" +
sexo + "' WHERE EMAIL='" + correo + "'";
        ot.executeSentence(sentenciaSQL);
    }
    if (!password.equals(null)) {
        String sentenciaSQL = "UPDATE USUARIO SET PASSWORD='" +
password + "' WHERE EMAIL='" + correo + "'";
        ot.executeSentence(sentenciaSQL);
    }
    if (!localidad.equals(null)) {
        String sentenciaSQL = "UPDATE USUARIO SET LOCALIDAD='" +
localidad + "' WHERE EMAIL='" + correo + "'";
        ot.executeSentence(sentenciaSQL);
    }
    if (!postal.equals(null)) {
        String sentenciaSQL = "UPDATE USUARIO SET CODIGO_POSTAL='" +
postal + "' WHERE EMAIL='" + correo + "'";
        ot.executeSentence(sentenciaSQL);
    }
    if (!direccion.equals(null)) {
        String sentenciaSQL = "UPDATE USUARIO SET DIRECCION='" +
direccion + "' WHERE EMAIL='" + correo + "'";
        ot.executeSentence(sentenciaSQL);
    }
}
}

```

3.6. COMPLETAR-PERFIL:

Servlet encargado de recibir la información actual sobre el usuario y mostrarla en las celdas oportunas cuando el usuario visita su perfil. Esto hace que el usuario pueda saber de manera sencilla sus datos, como el nombre, apellidos, dirección, contraseña, correo electrónico...

El encargado de obtener los datos es el método completarPerfil, que recibe como único parámetro el correo electrónico del usuario logeado en la página web actualmente.

Este se encarga de realizar tantas consultas tipo SELECT como datos del usuario haya en la base de datos, y los almacena todos en un arrayList de Strings. Este se le pasa al Servlet y posteriormente con el JSP correspondiente se muestran los valores sacados en las celdas correspondientes.

```

public ArrayList<String> completarPerfil(String correo) {
    ArrayList<String> datosUsuario = new ArrayList<String>();

    String sentenciaSQL = "SELECT NOMBRE FROM USUARIO WHERE EMAIL='" + correo + "'";
    ArrayList<String> res = ot.executeQueryReturn(sentenciaSQL);
    datosUsuario.add(res.get(0));

    sentenciaSQL = "SELECT APELLIDOS FROM USUARIO WHERE EMAIL='" + correo + "'";
    res = ot.executeQueryReturn(sentenciaSQL);
    datosUsuario.add(res.get(0));

    sentenciaSQL = "SELECT APELLIDOS FROM USUARIO WHERE EMAIL='" + correo + "'";
    res = ot.executeQueryReturn(sentenciaSQL);
    datosUsuario.add(res.get(0));

    sentenciaSQL = "SELECT SEXO FROM USUARIO WHERE EMAIL='" + correo + "'";
    res = ot.executeQueryReturn(sentenciaSQL);
    datosUsuario.add(res.get(0));

    sentenciaSQL = "SELECT PASSWORD FROM USUARIO WHERE EMAIL='" + correo + "'";
    res = ot.executeQueryReturn(sentenciaSQL);
    datosUsuario.add(res.get(0));

    sentenciaSQL = "SELECT LOCALIDAD FROM USUARIO WHERE EMAIL='" + correo + "'";
    res = ot.executeQueryReturn(sentenciaSQL);
    datosUsuario.add(res.get(0));

    sentenciaSQL = "SELECT CODIGO_POSTAL FROM USUARIO WHERE EMAIL='" + correo + "'";
    res = ot.executeQueryReturn(sentenciaSQL);
    datosUsuario.add(res.get(0));

    sentenciaSQL = "SELECT DIRECCION FROM USUARIO WHERE EMAIL='" + correo + "'";
    res = ot.executeQueryReturn(sentenciaSQL);
    datosUsuario.add(res.get(0));

    return datosUsuario;
}

```

3.7. COMPARAR:

Servlet encargado de mostrar por pantalla los hoteles que el usuario desea comparar.

```

public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    try {
        @SuppressWarnings("unchecked")
        ArrayList<ResultadoBusqueda> resultados = (ArrayList<ResultadoBusqueda>) request.getSession().getAttribute("resultados");
        ArrayList<ResultadoBusqueda> resultadosComparar = new ArrayList<ResultadoBusqueda>();
        if(resultados!=null){
            for (int i=0; i<resultados.size(); i++){
                String checkbox = request.getParameter(Integer.toString(i));
                if(checkbox!=null && checkbox.equals("OK")){
                    resultadosComparar.add(resultados.get(i));
                }
            }
        }
        request.getSession().setAttribute("resultadosComparar", resultadosComparar);
        response.sendRedirect("comparar.jsp");
    } catch (Exception e) {
        //En caso de error vamos pagina de error
        System.out.println("Error: " + e.getMessage());
        //response.sendRedirect("paginaError.html");
    }
}

```

Este consulta los resultados de la búsqueda y el valor de los checkbox correspondiente a cada hotel listado. Si está marcado indica que el usuario quiere compararlo, por lo tanto lo añade a un nuevo arrayList denominado resultadosComparar.

Una vez ejecutado todo el método devuelve el arrayList con todos los hoteles seleccionados para comparar, información que utiliza el JSP correspondiente para mostrarla por pantalla.

3.8. CONSULTAR:

Servlet encargado de recoger la información de la base de datos sobre la consulta de hoteles realizada por el usuario. Este tiene en cuenta la localidad seleccionada, la fecha de inicio y fin de la reserva y el número de personas.

```
public ArrayList<ResultadoBusqueda> hacerConsulta(String ciudad, int dia, int mes, int ano, int numPersonas){
    String tipo_habitacion = "familiar";
    switch(numPersonas){
        case 1 : tipo_habitacion = "individual"; break;
        case 2 : tipo_habitacion = "doble"; break;
        case 3 : tipo_habitacion = "triple"; break;
        case 4 : tipo_habitacion = "cuadruple"; break;
    }
    String sentenciaSQL = "SELECT DISTINCT hotel.nombre FROM HOTEL,localizacion,dispone,fecha WHERE " +
        "localizacion.ciudad='"+ciudad+"' AND " +
        "localizacion.id=hotel.localizacion_id AND " +
        "hotel.id=dispone.hotel_id AND " +
        "dispone.tipo_habitacion='"+tipo_habitacion+"' AND " +
        "dispone.fecha=fecha.id AND " +
        "fecha.dia='"+dia+"' AND fecha.mes='"+mes+"' AND fecha.ano='"+ano+"'";
    ArrayList<ResultadoBusqueda> resultados = new ArrayList<ResultadoBusqueda>();
    ArrayList<String> resultadosString = ot.executeQueryReturn(sentenciaSQL);
    int numResultados = resultadosString.size();
    while(numResultados > 0){
        sentenciaSQL = "SELECT hotel.nombre, estrellas, descripcion, tipo_habitacion, precio, hotel.id FROM hotel, localizacion, fecha, dispone WHERE " +
            "hotel.nombre='"+resultadosString.get(numResultados-1)+"' AND localizacion.id=hotel.localizacion_id AND " +
            "dispone.hotel_id=hotel.id AND " +
            "fecha.dia='"+dia+"' AND fecha.mes='"+mes+"' AND fecha.ano='"+ano+"'";
        ArrayList<String> res = ot.executeQueryReturn(sentenciaSQL);
        System.out.println(res.get(0) + " " + res.get(1) + " " + res.get(2) + " " + res.get(3));
        resultados.add(new ResultadoBusqueda(res.get(0),Integer.parseInt(res.get(1)),res.get(2),ciudad,
            dia, mes, ano, res.get(3),
            Integer.parseInt(ot.executeQueryReturn("SELECT COUNT(*) FROM DISPONE,HOTEL WHERE hotel.nombre='"+res.get(0)+"' AND dispone.hotel_id=hotel.id").get(0)),
            Double.parseDouble(res.get(4)),Integer.parseInt(res.get(5))));
        numResultados--;
    }
    return resultados;
}
```

Para obtener la información se llama al método hacerConsulta. Este realiza una consulta de tipo SELECT a la base de datos. Los resultados se almacenan en un arrayList de tipo String. Como nos interesa almacenar y mostrar toda la información de cada hotel, se realiza otra consulta SELECT por cada hotel añadido al arrayList anterior, sacando así datos como: el número de estrellas, número de habitaciones libres, información extra...

Toda esta información se le proporciona al JSP que es el encargado de mostrar por la pantalla los resultados.

3.9. ELIMINAR-USUARIO:

Servlet encargado de eliminar de la base de datos toda la información del usuario logeado actualmente.

```
public void eliminarUsuario(String email) {
    ot.executeSentence("DELETE FROM USUARIO WHERE EMAIL='"+email+"'");
}
```

Este realiza una sentencia de tipo DELETE en la base de datos con el correo electrónico del usuario.

3.10. RESERVAR:

Servlet encargado de añadir una nueva reserva a la base de datos con un correo electrónico y un hotel asignados.

Para ello se realiza una consulta de tipo INSERT en la tabla actividad, en la que se añade el mes y año de la reserva, junto al usuario, hotel y forma de pago.

```
public void reservar(String usuario, int id, int mes, int ano){
    Random r = new Random(); //Simula un posible generador de ids automaticas de cada reserva
    ot.executeSentence("INSERT INTO ACTIVIDAD(ID,USUARIO,HOTEL_ID,TIPO_HABITACION,FECHA1,FECHA2,METODO_PAGO)"
        + " VALUES (?,?,,?,?,,?)",
        r.nextInt(10000), usuario,id, "doble", mes, ano, "pago" );
}
```

4. METODOLOGÍA DE TRABAJO:

4.1. RECURSOS:

Durante el desarrollo de la práctica nos hemos apoyado en varios recursos para así facilitar el trabajo realizado. Se ha buscado bastante información sobre la programación de Servlets y JSP ya que era la primera vez que se realizaba esta tarea.

Además, se ha buscado información sobre la correcta configuración de los Servlets (directorio donde colocarlos dentro del servidor Tomcat, compilación de los archivos con las librerías necesarias, librerías necesarias para los Servlets...).

Se utilizó las clases dadas por los profesores de la asignatura Bases De Datos para facilitar la conexión con la base de datos Oracle y para reutilizar los métodos dados que permiten realizar consultas en ella (Selects, Insert, Delete...).

4.2. HERRAMIENTAS UTILIZADAS:

Para la programación de los Servlets se utilizó "Eclipse IDE for Java EE Developers" el cual es una herramienta perfecta para la creación de aplicaciones Web. Como editor de código (tanto para html, como jsp o java en algunos casos) se utilizó Sublime Text, ya que facilita en gran manera la programación y ayuda al usuario a no perder una gran cantidad de tiempo.

Para la Base de Datos se utilizó el gestor Oracle. Esto es debido a que se prefirió utilizar el gestor utilizado en la asignatura Base de Datos, pensando que nos facilitaría el trabajo una vez conocido el entorno. Además, los ficheros de conexión con la DB estaban preparados para conectarse a una DB gestionada con Oracle, y no con otros gestores.

Por último, para la creación del servidor Web se utilizó Tomcat, tal y como se recomendaba en las prácticas anteriores.

3.3. DISTRIBUCIÓN DEL TRABAJO:

La distribución de esta práctica se realizó de forma equitativa. Primero, durante las primeras sesiones de prácticas se investigó y busco información que facilitará la creación y programación de los Servlets, JSP y la conexión con la BD.

Una vez recopilada toda la información realizada, se procedió a programar un Servlet a modo de ejemplo para comprobar su funcionamiento. Tras realizar el Servlet a modo de ejemplo, se separaron entre los 2 miembros del grupo el resto Servlets.

Cuando todos estuvieron programados y probados, se procedió a juntarlos y realizar los JSP. Una vez terminado todo lo anterior, se procedió a comprobar el correcto funcionamiento de todos los archivos, que los resultados fuesen los correctos, que las reedirecciones funcionasen, y que se cumpliesen todos los objetivos marcados en el pdf entregado en la práctica.

Por último, se realizaron mejoras en el diseño (como listar los hoteles por orden de precio, de menor a mayor) y se facilitó la forma en la que el usuario recibe la información sobre cualquier error dado, como por ejemplo un registro con un correo electrónico en uso...

4. DIFICULTADES ENCONTRADAS:

Durante el desarrollo de la práctica nos encontramos con varias dificultades. En primer lugar se necesitaba encontrar una forma con la que conectar la base de datos y la página web. Para ello se buscó información por internet y en Moodle en la sección de la asignatura de Base De Datos. Al final se decidió usar el código proporcionado por los profesores de la asignatura mencionada, el cual permitía conectarse a una base de datos Oracle y realizar cualquier tipo de operación sobre ella.

La segunda dificultad fue la de programar los primeros Servlets ya que nunca antes se había realizado una tarea similar. Para informarnos utilizamos las diapositivas dadas en la clase de Sistemas de Información sobre la programación de Servlets y JSPs. Además, se tuvo bastantes problemas con la modificación del archivo XML en el que se indican las rutas de los distintos Servlets de la página web y con la colocación de estos dentro de las distintas carpetas del servidor web (Tomcat).

Por último se encontraron problemas en la programación de la SESSION dentro de la página web y el envío de los distintos atributos entre los Servlets y JSP. Para ello se buscó información por internet y se utilizaron distintos tutoriales para facilitar la tarea.

5. MANUAL DE INSTALACIÓN:

Para proceder a instalar la página web en un servidor se deben seguir los siguientes pasos (teniendo en cuenta que el servidor utilizado para el manual es Tomcat):

1. Una vez se dispone de la carpeta en la que están almacenados todos los archivos de la página web (tanto JSP como HTML o JAVA) se introduce dentro de la carpeta "webapps". En esta carpeta se pueden almacenar tantas páginas web como se deseen, mientras estas estén cada una en su carpeta correspondiente.
2. El usuario tiene que compilar los archivos JAVA almacenados en HotelLink/WEB-INF ejecutando el SH llamado compila.sh. Este archivo se encarga de compilar todas las clases (sin errores) utilizando los paquetes y librerías añadidos anteriormente.

3. Una vez realizados los pasos anteriores hay que proceder a encender el servidor. Para ello se va a la carpeta donde se tenga instalado el Tomcat y accedemos a la carpeta "bin". Dentro encontraremos varios archivos SH, los que necesitaremos serán startup.sh y shutdown.sh. El primero servirá para encender el servidor, mientras que el segundo lo apagará.
4. Cuando hayamos encendido el servidor la página web ya estará accesible. Si ponemos la ruta "localhost:8090/HotelLink" en cualquier navegador lo comprobaremos.

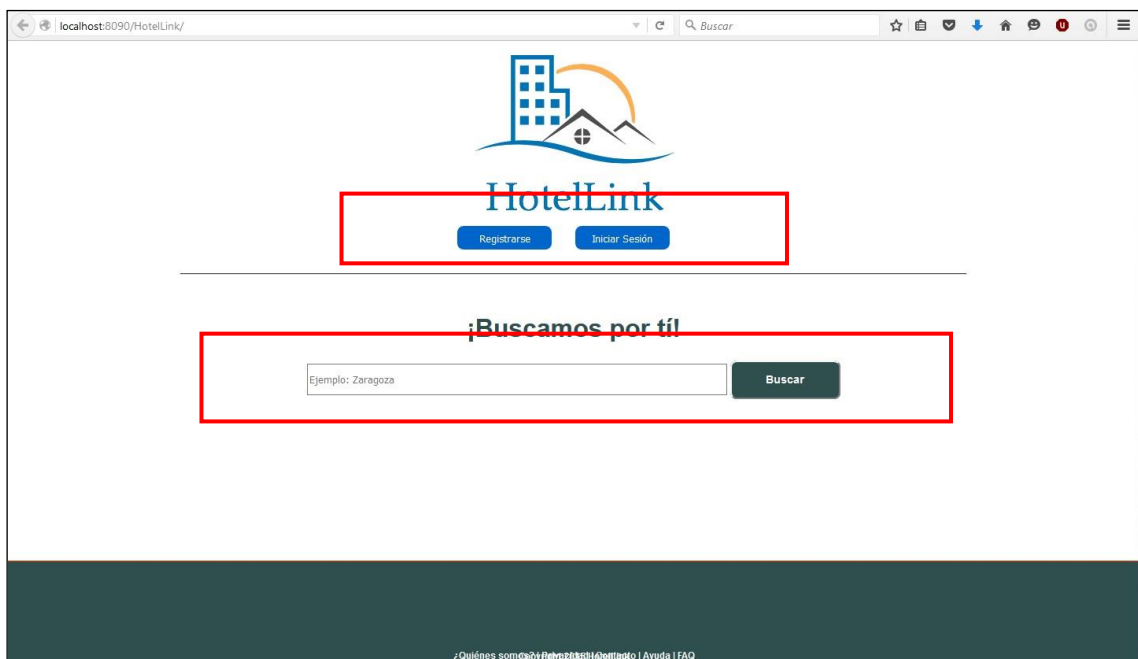
Si se desea realizar cualquier cambio en la página web se tendrán que seguir los siguientes pasos:

1. Si se desea modificar archivos JSP o HTML bastará con modificarlos con cualquier editor de texto y guardar los cambios.
2. Si se desea modificar un Servlet (JAVA) una vez modificado habrá que volver a compilar todos los archivos con el compila.sh explicado anteriormente. Si en vez que modificar un Servlet ya dado se quiere crear uno nuevo, basta con crear un nuevo archivo JAVA similar a los demás y guardarlo. Después se deberá añadir la ruta del nuevo Servlet en el archivo "web.xml" situado en la ruta HotelLink/WEB-INF.

6. MANUAL DE USUARIO:

Para entender de forma rápida el funcionamiento de la página web y aprender a usarla se recomienda al usuario leer el manual de usuario. Seguidamente se comentarán los puntos importantes a tener en cuenta:

- La página web por defecto se inicia en el index.jsp. En él se le da al usuario la opción de empezar una nueva búsqueda de hoteles o registrarse/iniciar sesión. Si el usuario no está registrado o no ha iniciado sesión podrá buscar



hoteles en nuestra base de datos, sin embargo no se le permitirá realizar ninguna reserva. Justo antes de reservar un hotel se le obligará al usuario a iniciar sesión.

- Si el usuario se ha registrado e inicia sesión tendrá a su disposición el “Perfil de Usuario”. En él aparecerán todos sus datos con los que se registró y se le permitirá modificar cualquiera de ellos excepto el correo electrónico. Dentro del perfil el usuario puede consultar sus últimas actividades, donde se le da la opción de marcar 2 fechas, una de inicio y otra de final, las cuales marcarán las fechas entre las que se quiere mostrar las reservas realizadas por el usuario facilitando así la búsqueda.
- Cualquier usuario que haya iniciado sesión podrá cerrarla dándole al botón “Cerrar Sesión”.
- Cualquier usuario que haya iniciado sesión podrá borrar su cuenta por completo de nuestra Base de Datos seleccionando la opción “Borrar Cuenta” que se encuentra dentro del perfil del usuario. Antes de borrar la cuenta se le pedirá al usuario una confirmación para así evitar una posible equivocación por parte de este.

localhost:8090/HotelLink/perfil.jsp

Buscar

HotelLink

Perfil Mis Actividades Cerrar Sesión Borrar Cuenta

Nombre:
Javier

Apellidos*:
JavierApellidos

Sexo:
☒ Hombre ☐ Mujer

Correo electrónico*:
javier@correo.com

Repetir correo electrónico*:

Contraseña:

Repetir contraseña*:

Dirección*:

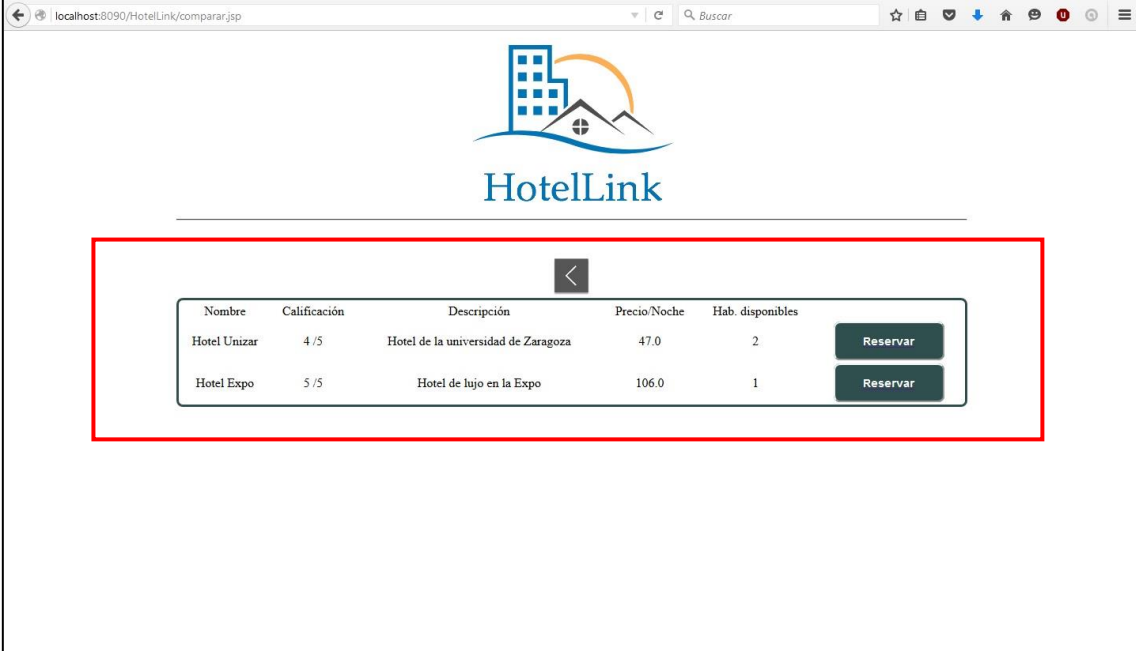
Localidad*:
Zaragoza

Código postal:

Nota: Los campos marcados con un asterisco deben rellenarse de forma obligatoria.

Enviar consulta

- A la hora de mostrar la lista de hoteles se le permitirá al usuario ordenar esto según el precio de menor a mayor. Esto facilita la búsqueda de hoteles en listas que son muy largas. Además el usuario dispone de una opción de comparar. En ellas se muestra la información de los hoteles seleccionados por el usuario de una manera más cómoda, permitiendo al usuario que de un vistazo pueda comprar precio y calidad de los hoteles en los que está interesado.



The screenshot shows a web browser window with the URL `localhost:8090/HotelLink/comparar.jsp`. The page features the HotelLink logo at the top. Below the logo, a red rectangular box highlights a comparison table of two hotels. The table has five columns: Nombre, Calificación, Descripción, Precio/Noche, and Hab. disponibles. Each row also includes a 'Reservar' button. The first row is for 'Hotel Unizar' with a rating of 4/5, a description 'Hotel de la universidad de Zaragoza', a price of 47.0, and 2 available rooms. The second row is for 'Hotel Expo' with a rating of 5/5, a description 'Hotel de lujo en la Expo', a price of 106.0, and 1 available room.

Nombre	Calificación	Descripción	Precio/Noche	Hab. disponibles	
Hotel Unizar	4 / 5	Hotel de la universidad de Zaragoza	47.0	2	<button>Reservar</button>
Hotel Expo	5 / 5	Hotel de lujo en la Expo	106.0	1	<button>Reservar</button>