

AD PRÁCTICA 3

Desarrollo de servicios web con SOAP

Estudiante

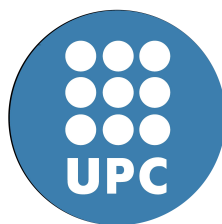
Alejandro Capella del Solar

Álex Torras González

Daniel Boté Mayer

Profesora

Silvia Llorente Viejo



26 de octubre de 2020

Índice general

Índice de figuras	3
I Introducción	4
0.1 De que trata la práctica	5
II Cuestionario	6
0.2 Apartado a	7
0.3 Apartado b	7
0.4 Apartado c	7
III Arquitectura del servicio web	8
1 Server	9
1.1 WS.java	9
1.2 Image.java	9
1.3 callsSQL.java	10
1.4 Operaciones del servicio	10
1.4.1 Register Image	10
1.4.2 Delete Image	10
1.4.3 List Image	10
1.4.4 MultiSearch	11
1.4.5 comprabaUser	11
2 Client_Application	12
2.1 Main.java	12
3 Servlet-Application	14
3.1 Servlet	14

3.1.1	RegistrarImagen.java	14
3.1.2	EliminarImagen.java	15
3.1.3	ModificarImagen.java	15
3.1.4	login.java	15
3.2	buscarImagen.java	15
3.3	RegistrarUsuario.java	16
3.4	logout.java	16
3.5	Páginas jsp	16
3.5.1	listImg.jsp	16
3.5.2	login.jsp	17
3.5.3	registrarImagen.jsp	17
3.5.4	eliminarImagen.jsp	17
3.5.5	menu.jsp	18
3.5.6	error.jsp	18
3.5.7	buscarImagen.jsp	19
3.5.8	RegistrarUsuario.jsp	19
3.5.9	registrarExito.jsp	20
3.5.10	logout.jsp	20
3.5.11	opciones.jsp	20
3.5.12	opcions_registrar.jsp	21
3.6	Aspectos a tener en cuenta	21

Índice de figuras

1	modificación modificarImagen	15
2	Error login	16
3	detalle código listImg	17
4	Formulario login	17
5	Formulario registrarImagen	18
6	Pantalla eliminar Imagen	18
7	Detalle formulario buscarImagen.jsp	20
8	Pantalla opciones.jsp	20
9	Confirmación registro Imagen	21

Parte I

Introducción

0.1. De que trata la práctica

En esta práctica hemos tenido que desarrollar una aplicación web y una aplicación con un servidor externo a los proyectos, por tanto, la base de datos por ejemplo no se ha podido acceder directamente, sino que se ha hecho a través del server.

Parte II

Cuestionario

0.2. Apartado a

- **¿Cuántos ficheros se han creado automáticamente en el cliente del servicio web?** Por una parte se han creado dos ficheros por cada operación implementada en el servidor. Estas son la operación y su respuesta. Además se ha creado una clase WS.java, una de servicio de WS que es WS_Service.java y otra que es package info. Estos ficheros que se han creado están guardados en una carpeta llamada Generated Sources. Por otra parte se han creado dos ficheros más en una carpeta que guarda los ficheros de configuración. La carpeta en cuestión es Configuration Files y dentro de esta carpeta, los ficheros son WS.wsdl y WS.xsd_1.xsd.

0.3. Apartado b

- **Consulta el fichero wsdl del servicio web que has creado accediendo a la URL `http://localhost:<puerto><NombreAplicacion><NombreServicio>?WSDL` e identifica las operaciones que ofrece el servicio web, sus parámetros y resultados.** Accediendo al link indicado, lo que se muestra es un listado con todas las operaciones implementadas. Aparece también para cada operación su request y su response. En la descripción de cada operación se informa que se trata de una operación SOAP y se identifican los parámetros mediante unas etiquetas. Las `<input></input>` hacen referencia a los parámetros que se le pasan a la función y `<output></output>` que retorna un parámetro.

0.4. Apartado c

- **Analiza el formato de la SOAP Request y SOAP response que se muestra cuando generamos un test con la opción Test Web Service de Netbeans (La opción está en botón derecho `<NombreAplicacion>/Web Services/<NombreServicio>`).** El request de la operación se hace en una página en la que tienes que pasar un parámetro como si fuera la función. Para nuestro caso resulta complicado ya que pasamos una clase entera. Una vez le das a testear la operación, redirige a una página donde indica cuáles son los parámetros de la función y cómo response. En nuestro caso nos ha saltado información de error indicando que el parámetro no era el correcto. El resultado correcto de un response es mostrar que ha pasado una vez se ha ejecutado la operación.

Parte III

Arquitectura del servicio web

Capítulo 1

Server

El servidor de esta práctica es el que se encarga de almacenar todas las imágenes que se registran tanto en la aplicación web como en la aplicación. Además, también se encarga de la gestión de la base de datos, es el que hace todas las consultas y accesos a ella. Por consiguiente cuando una de las dos aplicaciones necesita alguna cosa de la base de datos ha de acceder primero al servidor para conseguirla. Hemos creado tres paquetes para diferenciar las distintas clases. Un paquete lo hemos llamado basedatos y allí hemos añadido nuestra clase `callsSQL.java`. Otro paquete que hemos añadido es el de modelo donde está la clase `Image.java`. El último paquete creado es el de servicio y en él hemos metido nuestra clase `WS.java`.

1.1. WS.java

Esta clase contiene todas las operaciones necesarias que se vamos a llamar de forma remota. Es la clase que se conecta con nuestro DAO para acceder a la base de datos. Todas las operaciones que queramos hacer desde nuestra aplicación o desde nuestra aplicación web, en caso de querer acceder a la base de datos para hacer alguna acción con una imagen, deberán pasar primero por el WS para que este se comuniquen con el DAO y este con la base de datos. Esto está hecho así por un tema de consistencia, pues el objetivo es emular que nuestro servidor no está en la misma máquina que en la que estamos trabajando nosotros.

1.2. Image.java

Esta clase contiene todos los atributos necesarios para definir una imagen en la práctica. El motivo de la existencia de esta clase es que podamos pasar las imágenes y todo su contenido de forma correcta entre el servidor y la aplicación o la aplicación web.

1.3. callsSQL.java

Esta clase emula nuestro DAO, accede a través de llamadas de funciones a la base de datos para las diversas acciones que se pueden realizar con una imagen como registrarla, eliminarla o modificarla.

1.4. Operaciones del servicio

A continuación pasaremos a describir las operaciones implementadas en el servicio SOAP

1.4.1. Register Image

Esta es una de las operaciones que están definidas en nuestro servidor. Su función es recoger los datos provenientes de la imagen que proporcionan los clientes y registrarlos de forma correcta. La clase funciona de la siguiente manera: primero registra la imagen en disco y a continuación se comunica con el DAO para insertar la imagen en disco. En caso de que el insertado en la base de datos de un error, la imagen se elimina del disco y retorna un error.

1.4.2. Delete Image

La operación de DeleteImage es otra de las operaciones que esta implementada en nuestro servidor. La función de esta operación es la de eliminar la imagen que el cliente solicita tanto de la base de datos como del disco. Para eliminarla de la base de datos se comunicará con el DAO para que se él el que la elimina.

1.4.3. List Image

La operación listImage se ha encargado de coger todas las imágenes que se han registrado en el servidor, tanto las que provienen de la aplicación como las que vienen de la aplicación web y se las pasa al usuario. Toda la información de la imagen se la pasa al cliente a través de la clase Image. Esta clase guarda la información de los atributos relacionados con la imagen, como título o palabras claves, y además guarda el contenido de la imagen en un byteArray[] para que el cliente la pueda ver sin tener que descargarla.

1.4.4. MultiSearch

MultiSearch es la operación de búsqueda multicampo que sustituye a cada una de las búsquedas individuales del enunciado con el fin de reaprovechar al máximo el servlet y el jsp de la práctica 2. Simplemente hace uso del dao y realiza la consulta y devuelve al servlet que la invoca un listado de Image.

1.4.5. compruebaUser

Esta función se encarga de comprobar que un usuario existe en la base de datos. A la función le llega un usuario por parámetro y lo que hace es comunicarse con el DAO para que se este el que compruebe si la operación existe o no.

Capítulo 2

Client_Application

Otro apartado de la práctica que teníamos que desarrollar es una aplicación cliente. Para llevarla a cabo solo hemos creado un paquete donde hemos almacenado nuestra función main. La aplicación funciona desde consola y se pueden probar las distintas operaciones. Para verlas, cuando se ejecuta la aplicación se muestra un panel con un listado numérico. Una vez elegida la operación, redirige a la función específica en la que se describe paso a paso a través del terminal que pasos debe hacer el usuario. Hemos creado un servicio web de cliente que se conecta al servidor que hemos creado. Es por ello por lo que todas las operaciones que se realizan se mandan al servidor y es este el que ejecuta las acciones pertinentes tanto para las imágenes como para los usuarios. Para comprobar los usuarios o registrarlos, nos comunicamos con el servidor para que se acceda a la base de datos de este y compruebe si el usuario existe o no o que se registre. El motivo de hacerlo de esta forma es para que haya consistencia. Si se creará una base de datos local y se registra un usuario nuevo y añade una imagen, luego desde la aplicación web habrá imágenes con usuarios que no existen. De la forma en la que lo hemos implementado nosotros no. Si se registra un usuario, se podrá ver tanto desde la aplicación como desde la aplicación web.

2.1. Main.java

Es la clase principal donde se hacen todas las operaciones. En esta clase tenemos las funciones necesarias para comprobar que la aplicación funciona. La función main es la primera que se ejecuta cuando se arranca la aplicación. Esta es la que muestra a través de pantalla el listado con las distintas operaciones disponibles. Una vez seleccionada una de las posibles acciones, se redirige a cada una de las funciones correspondientes. Para el inicio de sesión de un usuario o para el registro de un usuario nos conectamos a el servicio web. Le pasamos los parámetros necesarios

y él se encarga de hacer las operaciones de registro o de inicio de sesión. Ambos casos son funciones booleanas que devuelve cierto si se la operación se ha hecho con éxito y falso en caso contrario. Al ser una aplicación que funciona desde terminal, se notifica al usuario que haga acciones concretas para que pueda funcionar todo de manera correcta. Un ejemplo de ello es en la función que usamos para registrar una imagen, se solicita al usuario que inserte en el terminal el path completo de la imagen para poder registrarla de manera correcta.

Capítulo 3

Servlet-Application

La última aplicación para desarrollar de la práctica ha sido una aplicación web como la de la práctica 2 con la diferencia que ahora no tenemos el servidor en la misma aplicación. La mayoría de las clases JSP se han quedado iguales pues solamente mostraban lo que había en el servlet. Las clases java en cambio si que se han tenido que modificar ya que estas son las encargadas de gestionar las acciones con las imágenes. Como ya hemos dicho, nuestro servicio no está alojado en la aplicación web sino que esta en otra parte. Para gestionar esto, hemos creado un servicio web de cliente que se conecta al servidor. Por lo tanto, igual que en la aplicación del cliente, nuestras operaciones relacionadas con las imágenes no se harán de forma local sino que se harán de forma remota. Para la gestión de usuarios en cambio, hemos creado un DAO local que se conecte a la base de datos para comprobar si los usuarios que quieren acceder a la aplicación existen o no. En esta aplicación web hemos creado dos paquetes, uno llamado basedatos donde hemos guardado nuestro DAO y otro paquete llamado servicio donde hemos guardado todas nuestras clases java.

3.1. Servlet

3.1.1. RegistrarImagen.java

En esta clase registramos una imagen, con su contenido y todos sus atributos. El sitio donde se guarda la imagen es desconocido para el usuario. Para guardar la imagen en vez de llamar a un DAO y que se comuniquen con la base de datos, llamamos a nuestro servicio web. Este se encargará de gestionar todo y será el que guarde la imagen, por lo tanto el cliente no tendrá acceso directo a la imagen. Para registrar la imagen hemos implementado la subida de archivos con `byteArray[]` de forma que todo el contenido de una imagen está en este vector y se pueda pasar



Figura 1: modificación modificarImagen

fácilmente.

3.1.2. EliminarImagen.java

Esta clase elimina las imágenes que un usuario ha registrado en su sesión. No lo hace de forma directa ya que estas están alojadas en otro directorio, por lo tanto no llama a un DAO para que elimine la imagen sino que llama a una operación del servicio web cliente que se encargue de ello. Este servicio web es el que tiene la imagen y en consecuencia el que realizara las operaciones con un DAO para eliminar la imagen de la base de datos y del disco.

3.1.3. ModificarImagen.java

La funcionalidad modificar imagen nos permite hacer un update de alguno de los campos que tienen las imágenes. Solo puede modificar la imagen el autor de dicha imagen y lo puede hacer desde la funcionalidad de buscar imagen o la de listar imagen. Hemos introducido algunos cambios de aspecto a esta funcionalidad.

3.1.4. login.java

Su función es la de comprobar que los datos que se introducen en el fichero JSP existen. Esta comprobación se hace mediante una llamada a la clase callsSQL, que es la que se comunica con la base de datos. Una vez se ha hecho la comprobación, si el usuario y la contraseña existen, la aplicación te redirige al menú. En caso de que no exista, te lleva a una página de error que te informa que el usuario o la contraseña están mal introducidos.

3.2. buscarImagen.java

Respecto a la práctica 2 hemos sustituido la instanciación del dao con su consulta, por la llamada remota a la operación multiseach, dejando toda la parte de

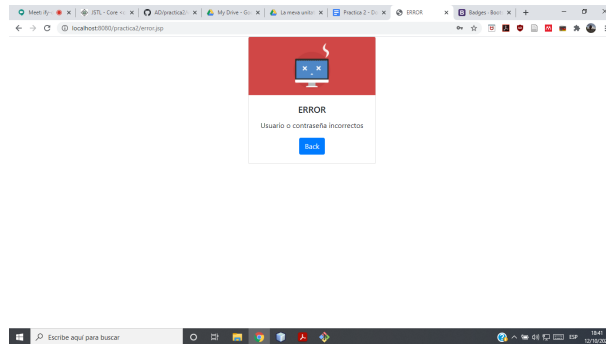


Figura 2: Error login

salida por pantalla del navegador. Recibimos de igual manera una lista de Image del servicio web

3.3. RegistrarUsuario.java

Esta clase comprueba que el usuario que se está intentando registrar no existe ya en la base de datos. En caso que exista manda un error y en caso contrario se inserta el nuevo usuario en la base de datos y se redirecciona hacia registrarExito.jsp. El usuario se registra a través de la operación `afegeixUser` contra el servicio web

3.4. logout.java

A esta clase se accede desde `logout.jsp`. Su función es cerrar la sesión del usuario actual y redireccionar otra vez hacia el login.

3.5. Páginas jsp

Las páginas JSP se han quedado igual excepto la de `listImg`.

3.5.1. listImg.jsp

Esta clase se encarga de listar las imágenes provenientes del servidor. Para hacerlo se comunica con este y coge todos los parámetros de la clase imagen. Este fichero no dispone de una carpeta de donde pueda coger las imágenes localmente, es por ello por lo que para pasar el contenido de forma remota a local hemos usado un `byteArray[]` que contiene todo el contenido de una imagen. La imagen la hemos conseguido mostrar por pantalla codificándola en base64 de la siguiente manera.

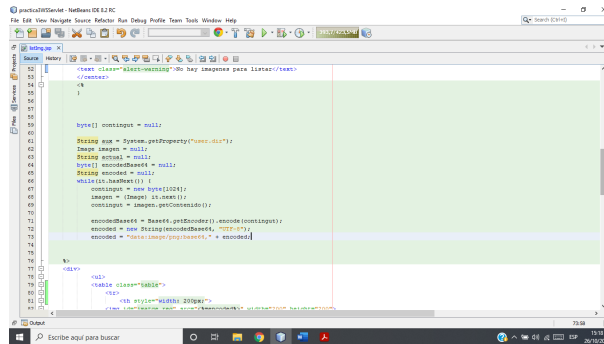


Figura 3: detalle código listImg

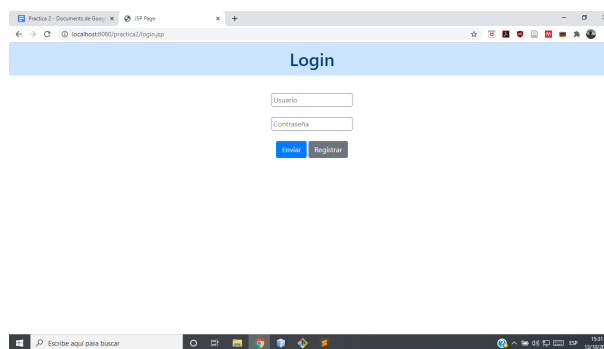


Figura 4: Formulario login

3.5.2. login.jsp

Este fichero es el primero que nos encontramos al abrir nuestra aplicación web. En este se tiene que pasar los parámetros de usuario y contraseña para poder acceder a la aplicación web. Comprobamos que existen pasando los parámetros al fichero java.

3.5.3. registrarImagen.jsp

Se ocupa de mostrar el formulario con toda la información de la nueva imagen que se quiere subir al servidor. Le pasa la info al servlet, con el método POST, de todos estos campos que están puestos como required.

3.5.4. eliminarImagen.jsp

A esta clase se llega solamente desde las opciones de listar imágenes o de buscar imágenes y su única función es la de preguntar si realmente quieres borrar el

Seleccionar archivo	Ningún archivo seleccionado
Títol	
Descripció	
Keywords	
Autor	
aaaa/mm/dd	
<input type="button" value="Submit"/>	

Figura 5: Formulario registrarImagen

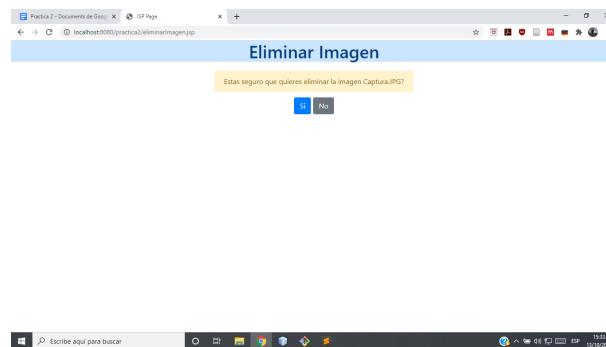


Figura 6: Pantalla eliminar Imagen

fichero o quieres volver atrás. Si das click a si, la página redirecciona a eliminarImagen.java. En caso contrario redirecciona al menú.

3.5.5. menu.jsp

Esta clase da tres opciones. Registrar imágenes, editarlas o eliminarlas. A parte, también es la clase a la que se vuelve de muchas otras clases después de haber hecho alguna acción.

3.5.6. error.jsp

Es la página de error. Es una jsp bastante sencilla. Consta de un componente de tarjeta de bootstrap y basándose en un atributo que recibe desde la excepción que pueda lanzar cualquier parte de la aplicación, imprimirá un botón y un mensaje que variará según el lugar de donde provenga la excepción para devolverte y te saldrá

un mensaje que te hablará de la clase de error que se ha producido. La codificación de los errores según el parámetro es la siguiente:

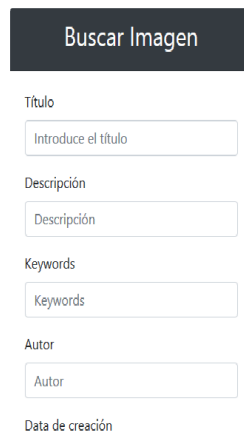
1. Error de SQL
2. Class not found Exception
3. No hay resultados de la búsqueda
4. Usuario o contraseña incorrectos
5. Error de entrada y salida
6. No se ha encontrado el fichero
7. Error en el servlet
8. Id de la imagen null
9. No se ha eliminado la base de datos
10. Ha fallado el registro de imagen
11. El usuario ya existe
12. Error inclasificable
13. Parámetros nulos

3.5.7. buscarImagen.jsp

Esta clase sigue igual. Esto es un formulario con una hoja de estilo que utiliza bootstrap. Tiene formato del componente card y también tiene un layout de tamaño medio para centrarlo en pantalla. Podría haber sido también una barra de búsqueda única, apenas habría cambiado algo el servlet pero hemos preferido ceñirnos al enunciado tal y como lo hemos entendido. El formulario utiliza el método GET para pasar la info al servlet dado que es una consulta que no modificará en ningún caso el estado de la base de datos, por lo que veremos aparecer los parámetros por la url.

3.5.8. RegistrarUsuario.jsp

Esta clase se encarga de pedir los parámetros necesarios para dar de alta a un usuario.



Formulario de búsqueda de imagen con el título "Buscar Imagen". El formulario contiene los siguientes campos de entrada:

- Título: Introduce el título
- Descripción: Descripción
- Keywords: Keywords
- Autor: Autor
- Data de creación: (campo vacío)

Figura 7: Detalle formulario buscarImagen.jsp

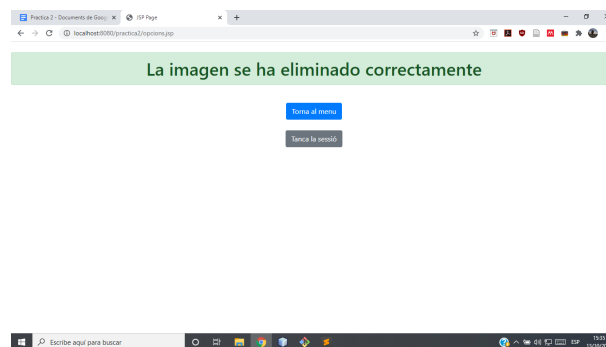


Figura 8: Pantalla opciones.jsp

3.5.9. registrarExito.jsp

Es una clase cuya única función es informar que el usuario se ha registrado con éxito en la base de datos y proporciona un botón para volver al login.

3.5.10. logout.jsp

Esta clase te pregunta si realmente quieres cerrar la sesión. Si la respuesta es "no", te lleva de vuelta al menú. En caso que sea "sí", redirecciona al fichero .java y de ahí al login.

3.5.11. opciones.jsp

Es una página que confirma que la imagen ha sido eliminada. correctamente sin más.

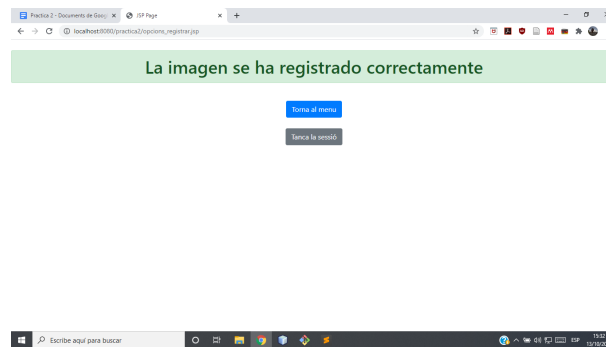


Figura 9: Confirmación registro Imagen

3.5.12. opciones_registro.jsp

Muestra que la imagen se ha registrado correctamente.

3.6. Aspectos a tener en cuenta

En nuestra práctica hemos considerado oportuno que toda gestión que tenga que ver con los usuarios desde la aplicación o desde la aplicación web, se haga a través del servicio web. Se ha hecho así por un tema de consistencia. Si se crea una base de datos local para la aplicación y otra para la aplicación web, lo que pasará es que habrá imágenes con usuarios que no existan en alguna de las dos aplicaciones.