Informe final prácticas 2 a 5

Alejandro Jesús Capella del Solar

Alex Torras Gonzalez

Daniel Boté Mayer

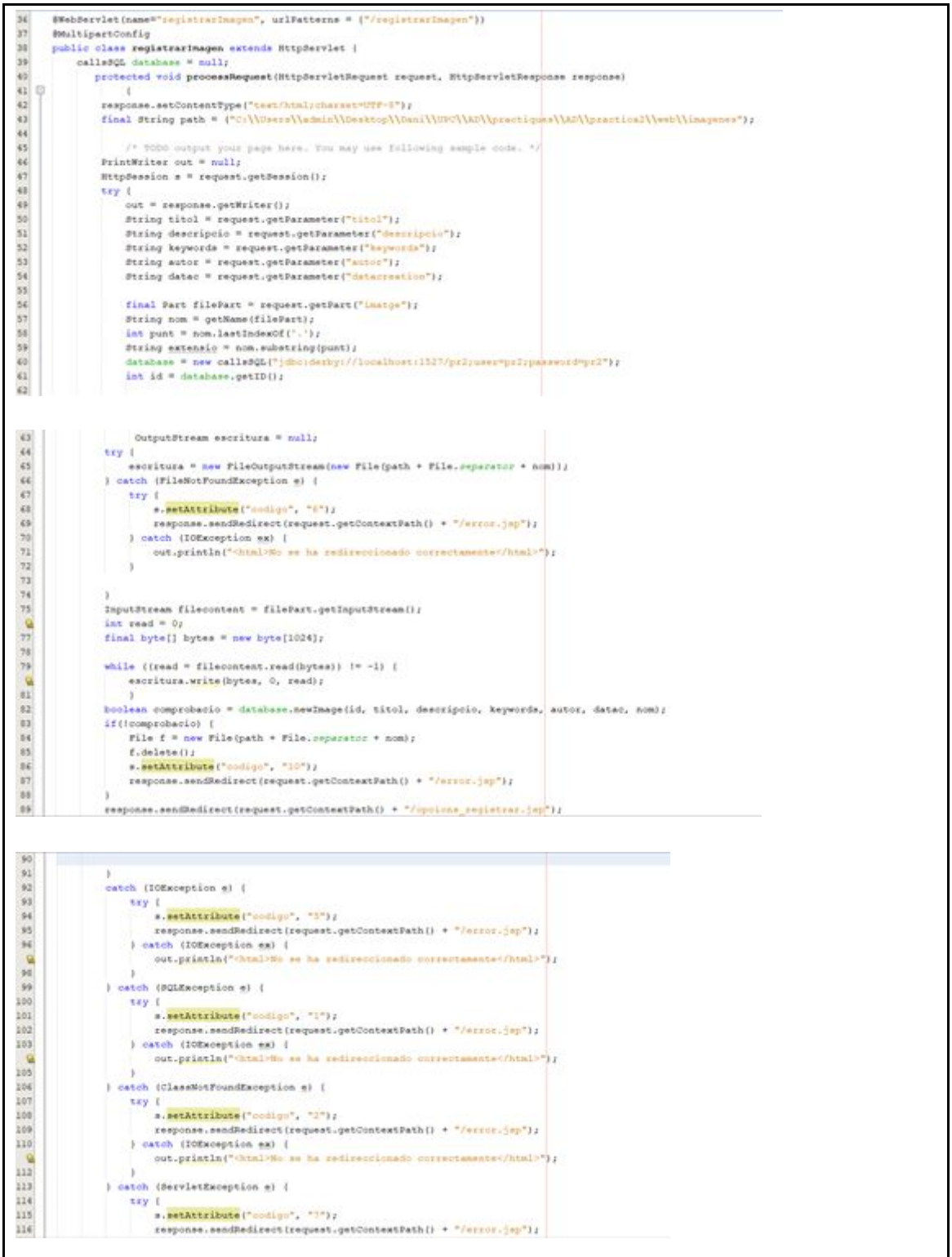
AD QT20-21

Índex

Contesta a las siguientes cuestiones referentes a las prácticas.

Práctica 2

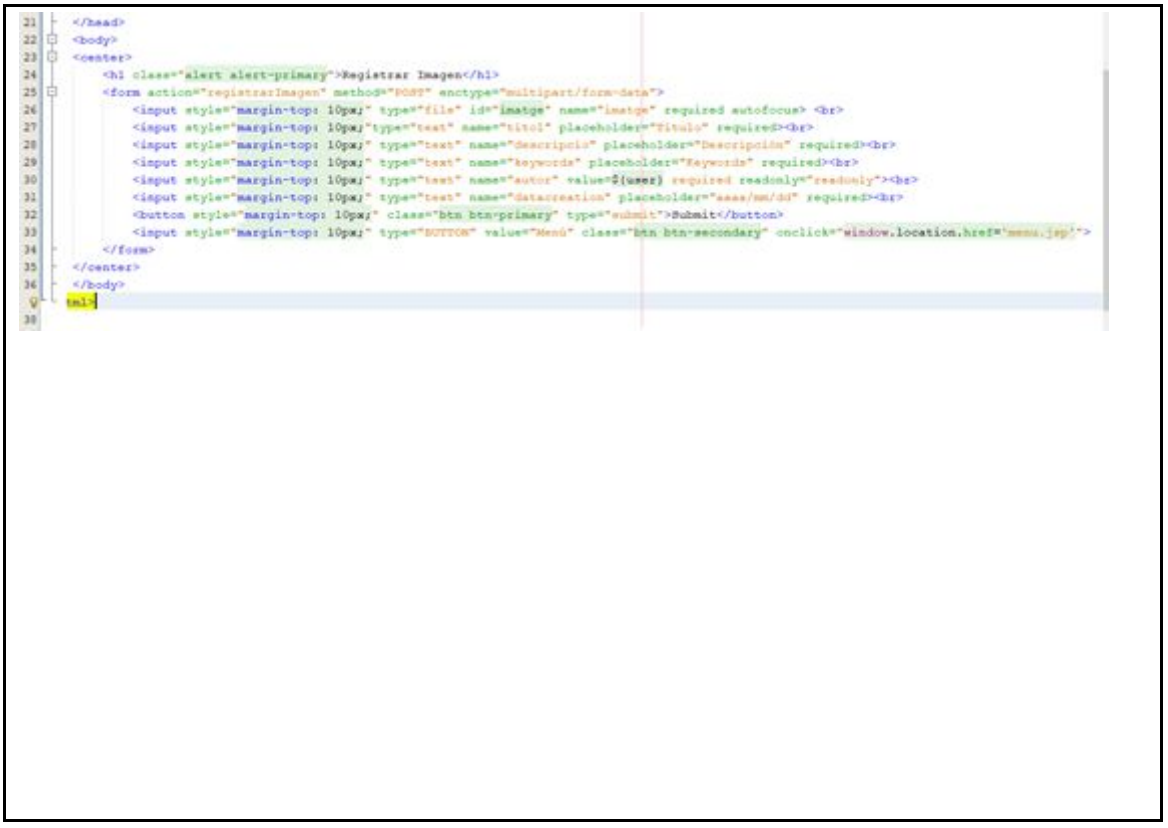
1. Copia en el cuadro el código del servlet que recoge los datos del formulario para registrar una imagen, guardarlos en la base de datos y almacenar el fichero con la imagen en disco.



```
@WebServlet(name="registrarImagen", urlPatterns = {"/registrarImagen"})
@MultipartConfig
public class registrarImagen extends HttpServlet {
    callsSQL database = null;
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            {
        response.setContentType("text/html;charset=UTF-8");
        final String path = ("C:\\Users\\admin\\Desktop\\Dani\\UPC\\AD\\practiques\\AD\\practica2\\web\\imagenes");

            /* TODO output your page here. You may use following sample code. */
        PrintWriter out = null;
        HttpSession s = request.getSession();
        try {
            out = response.getWriter();
            String titol = request.getParameter("titol");
            String descripcio = request.getParameter("descripcio");
            String keywords = request.getParameter("keywords");
            String autor = request.getParameter("autor");
            String datac = request.getParameter("datacreation");

            final Part filePart = request.getPart("imatge");
            String nom = getName(filePart);
            int punt = nom.lastIndexOf('.');
            String extensio = nom.substring(punt);
            database = new callsSQL("jdbc:derby://localhost:1527/pr2;user=pr2;password=pr2");
            int id = database.getID();

             OutputStream escritura = null;
        try {
            escritura = new FileOutputStream(new File(path + File.separator + nom));
        } catch (FileNotFoundException e) {
            try {
                s.setAttribute("codigo", "6");
                response.sendRedirect(request.getContextPath() + "/error.jsp");
            } catch (IOException ex) {
                out.println("<html>No se ha redireccionado correctamente</html>");
            }

        }
        InputStream filecontent = filePart.getInputStream();
        int read = 0;
        final byte[] bytes = new byte[1024];

        while ((read = filecontent.read(bytes)) != -1) {
            escritura.write(bytes, 0, read);
            }
        boolean comprobacio = database.newImage(id, titol, descripcio, keywords, autor, datac, nom);
        if(!comprobacio) {
            File f = new File(path + File.separator + nom);
            f.delete();
            s.setAttribute("codigo", "10");
            response.sendRedirect(request.getContextPath() + "/error.jsp");
        }
        response.sendRedirect(request.getContextPath() + "/opcions_registrar.jsp");

        }
        catch (IOException e) {
            try {
                s.setAttribute("codigo", "5");
                response.sendRedirect(request.getContextPath() + "/error.jsp");
            } catch (IOException ex) {
                out.println("<html>No se ha redireccionado correctamente</html>");
            }
        } catch (SQLException e) {
            try {
                s.setAttribute("codigo", "1");
                response.sendRedirect(request.getContextPath() + "/error.jsp");
            } catch (IOException ex) {
                out.println("<html>No se ha redireccionado correctamente</html>");
            }
        } catch (ClassNotFoundException e) {
            try {
                s.setAttribute("codigo", "2");
                response.sendRedirect(request.getContextPath() + "/error.jsp");
            } catch (IOException ex) {
                out.println("<html>No se ha redireccionado correctamente</html>");
            }
        } catch (ServletException e) {
            try {
                s.setAttribute("codigo", "3");
                response.sendRedirect(request.getContextPath() + "/error.jsp");
```

```
            } catch (IOException ex) {
                out.println("<html>No se ha redireccionado correctamente</html>");
            }
        }

        finally {
            try{
                Database.cerrarConexion();
            } catch (SQLException ex) {
                Logger.getLogger(registrarImagen.class.getName()).log(Level.SEVERE, null, ex);
            }
        }


          /* try {
               response.sendRedirect(request.getContextPath() + "/menu.jsp");
           } catch (IOException ex) {
               Logger.getLogger(registrarImagen.class.getName()).log(Level.SEVERE, null, ex);
           }*/

        }
```

2. Copia en el cuadro el código del formulario html que pide al usuario los datos de una imagen para registrarla.



```
</head>
<body>
<center>
    <h1 class="alert alert-primary">Registrar Imagen</h1>
    <form action="registrarImagen" method="POST" enctype="multipart/form-data">
        <input style="margin-top: 10px;" type="file" id="imatge" name="imatge" required autofocus> <br>
        <input style="margin-top: 10px;"type="text" name="titol" placeholder="Titulo" required><br>
        <input style="margin-top: 10px;" type="text" name="descripcio" placeholder="Descripción" required><br>
        <input style="margin-top: 10px;" type="text" name="keywords" placeholder="Keywords" required><br>
        <input style="margin-top: 10px;" type="text" name="autor" value="${user}" required readonly="readonly"><br>
        <input style="margin-top: 10px;" type="text" name="datacreation" placeholder="aaaa/mm/dd" required><br>
        <button style="margin-top: 10px;" class="btn btn-primary" type="submit">Submit</button>
        <input style="margin-top: 10px;" type="BUTTON" value="Menú" class="btn btn-secondary" onclick="window.location.href='menu.jsp'">
    </form>
</center>
</body>
tml>
```

Práctica 3

1. Copia en el cuadro la operación de registro de una imagen en SOAP.

```
/**
 * Web service operation
 */
@WebMethod(operationName = "RegisterImage")
public int RegisterImage(@WebParam(name = "image") Image image) {

    boolean salt = false;
    db = new callsSQL("jdbc:derby://localhost:1527/pr2;user=pr2;password=pr2");
    final String path = "C:\\Users\\admin\\Desktop\\Dani\\UPC\\AD\\practiques\\AD\\practica3\\practica3Server\\web\\imagenes";

    FileOutputStream ous = null;
    String nom = image.getFilename();
    byte[] contingut = image.getContenido();

    File fo = null;
    try {
        //fi = new File(nom);
        fo = new File(path + File.separator + nom);
        ous = new FileOutputStream(fo);
        ous.write(contingut);
        image.setId(db.getID());
        boolean comprobacio = db.newImage(image.getId(), image.getTitol(), image.getDescripcio(), image.getKeywords(), image.getAutor(), image.getDatac(), image.getFilename());
        if(!comprobacio) {
            File f = new File(path + File.separator + nom);
            f.delete();
            return 0;
        }

    } catch (FileNotFoundException ex) {
        codi_error = 6;
        System.out.println("La causa és: " + ex.getCause());
        salt = true;
    } catch (IOException e) {
        codi_error = 5;
        System.out.println("La causa de error es de IO y el motivo: " + e.getCause());
        salt = true;
    } catch (SQLException e) {
        codi_error = 1;
        System.out.println("La causa del error es relativa a la base de datos y el motivo: " + e.getCause());
        salt = true;
    } finally {
        try {
            db.cerrarConexion();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if(salt) return 0;
    else return 1;
}
```

2. Copia en el cuadro la operación de búsqueda por título en SOAP.

```
/**
 * Web service operation
 */
@WebMethod(operationName = "SearchbyTitle")
public List SearchbyTitle(@WebParam(name = "title") String title) {
    //TODO write your implementation code here:
List<Image> resultados = null;
    try{

    db = new callsSQL("jdbc:derby://localhost:1527/pr2;user=pr2;password=pr2");
    resultados = db.buscarImagenporTitulo(title);

     } catch (SQLException e) {
         codi_error = 1;
         System.out.println("La causa del error es: " + e.getCause());
        //e.printStackTrace();

        // lasesion.setAttribute("codigo", "1");

    }

    finally
    {
        try {
            db.cerrarConexion();

        } catch (SQLException ex) {
            ex.printStackTrace();
        }

    }
 return resultados;
}
```

El cliente desarrollado.

Hemos desarrollado un cliente de SOAP. El proceso ha sido básicamente coger nuestro cliente de SOAP, eliminar las funciones sobrantes y adaptarlo.

Las principales diferencias entre su servidor y el nuestro es que:

-Sus operaciones SOAP devuelven List<imagen> mientras que las nuestras devuelven List.

-Su servidor SOAP solo dispone de las siguientes operaciones:

- DeleteImage
- ListImages
- ModifyImage
- RegisterImage
- SearchbyAuthor

- SearchbyCreaDate
- SearchbyId
- SearchbyKeyWords
- SeachbyTitle
- getNewImageId
- userExists

-Tienen una clase llamada usuario compuesta de usuario y contraseña pero nosotros no la usaremos.

- El equivalente a nuestra clase Image es su clase imagen, pero su clase imagen tiene datos en forma de String en vez de que nosotros los tenemos en forma de bytes. El campo tiene el siguiente nombre:

- String encodedData;

Hemos decidido no repetir la documentación del cliente puesto que es básicamente el mismo que nuestro cliente de SOAP de la práctica 3 adaptado y sin las funcionalidades de más que nosotros teníamos. Han cambiado los nombres de los métodos de la clase imagen y por lo demás ha sido suficiente con poner imagen allá donde ponía Image.Otra cosa con la que hemos tenido que lidiar es que muchas partes de la práctica estaban preparadas para funcionar con un sistema Linux. También ha habido un problema de permisos derivados del trabajo de la otra pareja en sistemas linux. Pero su servidor da un error que no entendemos: Client received SOAP Fault from server: javax.ejb.EJBException

Quisimos desarrollar un cliente REST pero este servidor rest apenas es un servicio SOAP encapsulado en uno REST, es decir, devuelve un String con la lista de resultados de las consultas separados por espacio, cada operación solo es una llamada a un servicio SOAP, es un servidor multitier en el peor sentido, pero no devuelve páginas html así que un cliente con un servidor REST así llevaría mucho trabajo.

4)

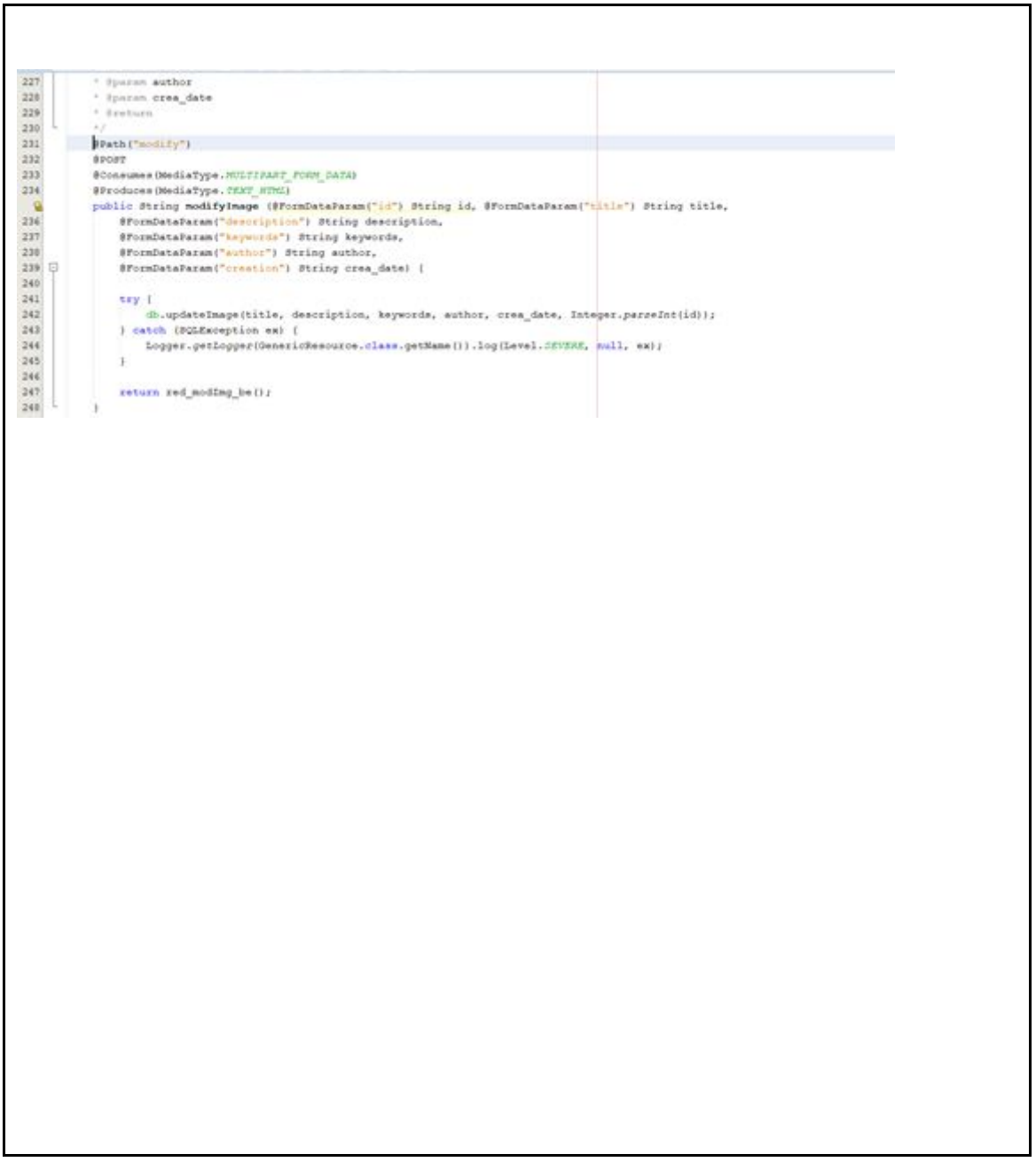
Hemos añadido las búsquedas individuales ,aparte de la ya mencionada búsqueda multicampo, que faltaban en la práctica 3 y 4. El código está en el zip entregado.

3. Copia en el cuadro el código que llama a una de las operaciones del servicio web de imágenes en SOAP.

```
public class modificarImagen extends HttpServlet{

    @WebServiceRef(wsdlLocation = "WEB-INF/wsdl/localhost_8080/practica3Server/WS.wsdl")
    private WS_Service service;
    Image imagen = new Image();

    protected void processRequest(HttpServletRequest request, HttpServletResponse response) {
        response.setContentType("text/html;charset=UTF-8");
        HttpSession s = request.getSession();


            /* TODO output your page here. You may use following sample code. */
            String titol = request.getParameter("titol");
            String descripcio = request.getParameter("descripcio");
            String keywords = request.getParameter("keywords");
            String autor = request.getParameter("autor");
            String datac = request.getParameter("datac");
            Integer id = Integer.parseInt(request.getParameter("id"));

            imagen.setTitol(titol);
            imagen.setDescripcio(descripcio);
            imagen.setKeywords(keywords);
            imagen.setAutor(autor);
            imagen.setDatac(datac);
            imagen.setId(id);

            servicio.WS_Service service = new servicio.WS_Service();
            servicio.WS port = service.getWSPort();
            int retorno = port.modifyImage(imagen);
            if (retorno == 0) {
            try {
                response.sendRedirect(request.getContextPath() + "/opcions_modificar.jsp");
            } catch (IOException ex) {
                Logger.getLogger(modificarImagen.class.getName()).log(Level.SEVERE, null, ex);
            }
            }
            else {
            try {
                s.setAttribute("codigo", "10");
                response.sendRedirect(request.getContextPath() + "/error.jsp");
            } catch (IOException ex) {
                Logger.getLogger(modificarImagen.class.getName()).log(Level.SEVERE, null, ex);
            }
            }

    }
```

Práctica 4

1. Copia en el cuadro la operación para modificar una imagen ya existente en REST.



```
     * @param author
     * @param crea_date
     * @return
     */
    @Path("modify")
    @POST
    @Consumes(MediaType.MULTIPART_FORM_DATA)
    @Produces(MediaType.TEXT_HTML)
    public String modifyImage (@FormDataParam("id") String id, @FormDataParam("title") String title,
        @FormDataParam("description") String description,
        @FormDataParam("keywords") String keywords,
        @FormDataParam("author") String author,
        @FormDataParam("creation") String crea_date) {

        try {
            db.updateImage(title, description, keywords, author, crea_date, Integer.parseInt(id));
        } catch (SQLException ex) {
            Logger.getLogger(GenericResource.class.getName()).log(Level.SEVERE, null, ex);
        }

        return red_modImg_be();
    }
```

2. Copia en el cuadro la operación para buscar una imagen por palabra clave en REST.

```
/**
* GET method to search images by keyword
* @param keywords
* @return
*/
@Path("searchKeywords/{keywords}")
@GET
@Produces(MediaType.TEXT_HTML)
public String searchByKeywords (@PathParam("keywords") String keywords) {
db = new callsSQL("jdbc:derby://localhost:1527/pr2;user=pr2;password=pr2");
    List<Image> resultados = null;
    String a = null;
    String b = null;
    String c = null;
    try {
        resultados = db.buscarImagenporKeywords(keywords);
        if (resultados != null) {
            a = "<!DOCTYPE html>\n"
                    + "<html>\n"
                    + "<head>\n"
                    + "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\">\n"
                    + "<title> Resultat </title>\n"
                    + "<link rel=\"stylesheet\" href=\"https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css\" integrity=\"sha384-J
                    + "</head>\n"
                    + "<body>\n"
                    + "<H1>Resultado de la búsqueda</H1></br>\n"
                    + "<td><a style=\"float: right\" class=\"btn btn-primary btn-lg active\" onClick=\"history.go(-2);\" role=\"button\" aria-pressed=\
                    + "<table>\n"
                    + "<thead>\n"
                    + "<tr>\n"
                    + "<th scope=\"col\">id</th>\n"
                    + "<th scope=\"col\">title</th>\n"
                    + "<th scope=\"col\">description</th>\n"
                    + "<th scope=\"col\">keywords</th>\n"
                    + "<th scope=\"col\">author</th>\n"
                    + "<th scope=\"col\">creation_date</th>\n"
                    + "<th scope=\"col\">storage_date</th>\n"
                    + "<th scope=\"col\">filename</th>\n"
                    + "</tr>\n"
                    + "</thead>\n"
                    + "<tbody>\n";

                for (Image i : resultados) {
                a += "<tr>\n"
                        + PrintImageData(i)
                        + "</tr>\n";
                }

                c = "</tbody>\n"
                        + "</table>\n"
                        + "</body>\n"
                        + "</html>\n";

                a.concat(c);
            } else {
                a = error("3");
            }

        } catch (SQLException ex) {
            Logger.getLogger(GenericResource.class.getName()).log(Level.SEVERE, null, ex);

        }
        return a;
    }

    private String PrintImageData(Image im) {
        String a
                = "<th scope=\"row\" value=\"" + im.getId() + "\"> " + im.getId() + " </th>\n"
                + "<td name=\"titol\" value=\"" + im.getTitol() + "\">" + im.getTitol() + "</td>\n"
                + "<td name=\"descripcio\" value=\"" + im.getDescripcio() + "\"> " + im.getDescripcio() + " </td>\n"
                + "<td name=\"keywords\" value=\"" + im.getKeywords() + "\">" + im.getKeywords() + "</td>\n"
                + "<td name=\"autor\" value=\"" + im.getAutor() + "\">" + im.getAutor() + "</td>\n"
                + "<td name=\"datac\" value=\"" + im.getDatac() + "\"> " + im.getDatac() + "</td>\n"
                + "<td name=\"datas\" value=\"" + im.getDatas() + "\"> " + im.getDatas() + "</td>\n"
                + "<td name=\"nom\" value=\"" + im.getFilename() + "\"> " + im.getFilename() + "</td>\n";
        return a;
    }
```

3. Copia en el cuadro el código que llama a una de las operaciones del servicio web de imágenes en REST.

```
<%@page import="java.util.Base64"%>
<%@page import="servicio.Image"%>
<%@page import="java.util.Iterator"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <%
    final String path = "http://localhost:8080/RestAD/";
    %>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">
        <title>JSP Page</title>
    </head>
    <body>
<CENTER>
        <h1 class="alert alert-primary">Modificar Imagen</h1>
        <form action="<%=path%>webresources/generic/modify"  method="POST" enctype="multipart/form-data">
            <input style="margin-top: 10px;" type="text" name="id" placeholder="Insertar id Imagen" required autofocus>
            <br>
            <input style="margin-top: 10px;" type="text" name="filename" placeholder="Insertar nombre Imagen" required autofocus>
            <br>
            <input style="margin-top: 10px;" type="text" name="title" placeholder="Titulo" required autofocus>
            <br>
            <input style="margin-top: 10px;" type="text" name="description" placeholder="Descripción" required>
            <br>
            <input style="margin-top: 10px;" type="text" name="keywords" placeholder="Keywords" required>
            <br>
            <input style="margin-top: 10px;" type="text" name="author" placeholder="Autor" required>
            <br>
            <input style="margin-top: 10px;" type="text" name="creation" placeholder="aaaa/mm/dd" required>
            <br>
            <button style="margin-top: 10px;" class="btn btn-primary" type="submit">Submit</button>
            <input style="margin-top: 10px;" type="BUTTON" value="Menú" class="btn btn-secondary" onclick="window.location.href='menu.jsp'">
        </form>
    </CENTER>
    </body>
</html>
```

Práctica 5

Compara los siguientes aspectos de la funcionalidad desarrollada en las prácticas 2, 3 y 4.

1. Facilidad de implementación de la parte cliente y la parte servidor.

La implementación de la práctica 2 tanto la parte de servidor como la de cliente resultó fácil ya que lo implementamos todo dentro de un mismo proyecto.

Realizar la práctica 3 nos resultó medianamente fácil implementar tanto el cliente como el servidor.

La práctica 4 fue compleja de implementar la parte del cliente. La parte del servidor se pudo implementar con bastante facilidad ya que lo único complejo era crear los metodos REST y Netbeans los montaba fácil. La complejidad en la parte del cliente fue la de controlar la sesión, que al final no se pudo y volver del servidor al cliente.

2. Tiempo de respuesta para la funcionalidad de registro de imagen. Para poder realizar la comparación, comenta la parte de upload de la página en la Práctica 2.

Practica 2: 54ms

Practica 3: 246.09ms

Práctica 4: 67ms

3. Compara el formato de las peticiones y las respuestas en SOAP y REST. ¿Cómo se realiza el envío de objetos complejos como por ejemplo las listas en ambos servicios?

En SOAP se devuelve la propia lista con toda la información para que sea el cliente el que trate esos datos y los muestre en el html. En cambio, en REST es el propio servidor el que devuelve la página html en forma de String.

Todas las prácticas

1. Detalla las ampliaciones que hayas realizado en cada práctica. Algunos ejemplos de ampliaciones son: funcionalidades extra de gestión de imágenes, jsp para gestión de errores, funciones extra de búsqueda, etc. Puedes copiar el código correspondiente a cada ampliación.

A continuación detallamos las ampliaciones hechas en cada práctica.

En la práctica 2 hicimos una serie de ampliaciones. Las dos principales han sido el registro de usuarios y el logout. Además creamos ficheros JSP de información cuando se realizaba una operación. En concreto estos ficheros JSP informan cuando una imagen se registra con éxito, cuando un usuario se registra con éxito o cuando una imagen se elimina con éxito.

```
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384
        <title>JSP Page</title>
    </head>
    <body>
    <CENTER>
        <br>
        <h1 class="alert alert-success"> La imagen se ha eliminado correctamente </h1>
        <br>
        <input type="BUTTON" value="Volver al menú" class="btn btn-primary" onclick="window.location.href='menu.jsp'">
        <br>
        <br>
        <input type="BUTTON" value="Cerrar la sesión" class="btn btn-secondary" onclick="window.location.href='logout.jsp'">
    </CENTER>
    </body>
</html>
```

Imagen eliminada correctamente

Imagen registrada correctamente

Usuario registrado correctamente

La búsqueda por múltiples campos arrastraba un bug de resultados no deseados que como veremos en la práctica 3 y 4 solucionamos, con lo que ahora funciona de manera correcta en prácticas 2,3 y 4.

Práctica 3:

En esta práctica también desarrollamos una serie de funcionalidades adicionales.

Dos de ellas son las que se pidieron como trabajo adicional, en concreto la subida de imágenes con SOAP y la búsqueda combinada.

A parte de estas dos operaciones extras, introducimos los ficheros JSP mencionados previamente para los avisos de que se ha registrado un usuario, una imagen o se ha eliminado una imagen correctamente. Además añadimos otro fichero JSP que se muestra para informar que una imagen se ha modificado correctamente.

Como último detalle recalcamos que también está implementada la búsqueda por diversos campos, esta vez ya debugada y funcionando correctamente. El núcleo de la búsqueda por diversos campos se halla en la clase CALLSSQL donde se monta la consulta bajo demanda según el número de campos introducidos tal y como se muestra en los siguientes pantallazos.

```java
private List<String> buildParamString() {
    List<String> paramName = new ArrayList<>();
    paramName.add("title");
    paramName.add("description");
    paramName.add("keywords");
    paramName.add("author");
    paramName.add("creation_date");
    paramName.add("storage_date");
    paramName.add("filename");
    return paramName;
}

private List<String> getListParamValue(String titol, String descripcio, String keywords, String autor, String datac, String datas, String filename) {
    List<String> paramValue = new ArrayList<>();
    paramValue.add(titol);
    paramValue.add(descripcio);
    paramValue.add(keywords);
    paramValue.add(autor);
    paramValue.add(datac);
    paramValue.add(datas);
    paramValue.add(filename);
    return paramValue;
}
```

```java
private List<Boolean> getListParam(String titol, String descripcio, String keywords, String autor, String datac, String datas, String filename

    List<Boolean> listParam;
    listParam = new ArrayList<>();
    if (titol != null && !titol.isEmpty()) {
        listParam.add(true);
    } else {
        listParam.add(false);
    }
    if (descripcio != null && !descripcio.isEmpty()) {
        listParam.add(true);
    } else {
        listParam.add(false);
    }
    if (keywords != null && !keywords.isEmpty()) {
        listParam.add(true);
    } else {
        listParam.add(false);
    }
    if (autor != null && !autor.isEmpty()) {
        listParam.add(true);
    } else {
        listParam.add(false);
    }
    if (datac != null && !datac.isEmpty()) {
        listParam.add(true);
    } else {
        listParam.add(false);
    }
    if (datas != null && !datas.isEmpty()) {
        listParam.add(true);
    } else {
        listParam.add(false);
    }
    if (filename != null && !filename.isEmpty()) {
        listParam.add(true);
    } else {
        listParam.add(false);
    }

    return listParam;
```

```
private String BuildQuery(List<Boolean> listParam, List<String> paramName) {
    String consulta;
    int j = 0;
    String nameParam;
    consulta = "SELECT * from IMAGE ";
    int tope = 8;
    int topetemporal = 0;
    for (Boolean i : listParam) {
        if (i == true) {
            tope = topetemporal;
        }
        topetemporal++;
    }
    if (tope < 8)//hay parametros
    {
        consulta = consulta + "where ";
        for (Boolean i : listParam) {
            if (i == true) {
                consulta = consulta + "(" + paramName.get(j) + " like '%'||?||'%')\n";
                if (j < tope)
                    consulta = consulta + "OR ";
            }


            j++;
        }
    }
    return consulta;
}
```

```
public List<Image> buscarImagen(String titol, String descripcio, String keywords, String autor, String datac, String datas, String filename) throws SQ
    List<Image> bilers;
    bilers = new ArrayList<>();
    ResultSet rs = null;
    ResultSet rs2 = null;
    PreparedStatement statement = null;
    //PreparedStatement statement2 = null;


    //try {
    boolean ok;
    //String consulta2;
    List<Boolean> listParam = null;
    List<String> paramName = null;
    List<String> paramValue = null;
    listParam = getListParam(titol, descripcio, keywords, autor, datac, datas, filename);
    paramName = buildParamString();
    paramValue = getListParamValue(titol, descripcio, keywords, autor, datac, datas, filename);
    String consulta = BuildQuery(listParam, paramName);
    //consulta2 = BuildQueryNumber(listParam,paramName);
    /*"SELECT * from IMAGE where (title like '%?%')"
            + " OR (description like '%?%')"
            + " OR (keywords like '%?%')"
            + " OR (author like '%?%')"
            + " OR (creation_date like '%?%')"
            + " OR (storage_date like '%?%')"
            + " OR (filename like '%?%')";
     */
    //statement2 = cn.prepareStatement(consulta2);
    statement = cn.prepareStatement(consulta);
    int j = 1;
    for (int i = 0; i < 7; i++) {
        if (listParam.get(i) == true) {//parámetros activos
            statement.setString(j,paramValue.get(i));
            //statement2.setString(j, paramValue.get(i));
            j ++;
        }
    }
    /*
    statement.setString(1, titol);
    statement.setString(2, descripcio);
    statement.setString(3, keywords);
```

```
        statement.setString(4, autor);
        statement.setString(5, datac);
        statement.setString(6, datas);
        statement.setString(7, filename);
/
        rs = statement.executeQuery();
        //rs2= statement2.executeQuery();
        if(!HayParam(listParam))

        {
        bilers = null;
        }
        else
        {
        while (rs.next()) {
            Image biler = new Image();
            biler.setId(rs.getInt("id"));
            biler.setTitol(rs.getString("title"));
            biler.setDescripcio(rs.getString("description"));
            biler.setKeywords(rs.getString("keywords"));
            biler.setAutor(rs.getString("author"));
            biler.setDatac(rs.getString("creation_date"));
            biler.setDatas(rs.getString("storage_date"));
            biler.setFilename(rs.getString("filename"));
            ok = bilers.add(biler);
            //
        }
        }

        return bilers;
```

Y la operación en SOAP queda definida así:

```
/**
 * Web service operation
 */
@WebMethod(operationName = "MultiSearch")
public List MultiSearch(@WebParam(name = "titulo") String titulo, @WebParam(name = "description") String description, @WebParam(name = "keywords") Str
    List<Image> resultados = null;
    try{

    db = new callsSQL("jdbc:derby://localhost:1527/pr2;user=pr2;password=pr2");
    resultados = db.buscarImagen(titulo, description, keywords, autor, datacreation, datasubida, filename);

     } catch (SQLException e) {
         codi_error = 1;
         System.out.println("La causa del error es: " + e.getCause());
        //e.printStackTrace();

        // lasesion.setAttribute("codigo", "1");

    }

    finally
    {
        try {
            db.cerrarConexion();

        } catch (SQLException ex) {
            ex.printStackTrace();
        }

    }
 return resultados;
}
```

En la práctica 4 a parte de de usar todas las funcionalidades extras de las prácticas anteriores, implementamos también la funcionalidad de búsqueda combinada ya que reutilizamos la clase que hacía esto mismo en la práctica 3 como hemos indicado anteriormente. En REST hemos definido la operación de la siguiente manera:

```
@Path("MultiSearch")
@GET
@Consumes(MediaType.MULTIPART_FORM_DATA)
@Produces(MediaType.TEXT_HTML)
public String MultiSearch(@QueryParam("titol") String titol, @QueryParam("descripcio") String descripcio, @QueryParam("keywords") String keywords, @Qu
    db = new callsSQL("jdbc:derby://localhost:1527/pr2;user=pr2;password=pr2");
    List<Image> resultados = null;
    String a = null;
    try {
        resultados = db.buscarImagen(titol, descripcio, keywords, autor, datac, datas, filename);
        if (resultados != null) {
            a = "<!DOCTYPE html>\n"
                    + "<html>\n"
                    + "<head>\n"
                    + "<title> Resultat </title>\n"
                    + "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\">\n"
                    + "<link rel=\"stylesheet\" href=\"https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css\" integrity=\"sha384-J
                    + "</head>\n"
                    + "<body>\n"
                    + "<H1>Resultado de la busqueda</H1></br>\n"
                    + "<td><a style=\"float: right\" class=\"btn btn-primary btn-lg active\" onClick=\"history.go(-2);\" role=\"button\" aria-pressed=
                    + "<table>\n"
                    + "<thead>\n"
                    + "<tr>\n"
                    + "<th scope=\"col\">id</th>\n"
                    + "<th scope=\"col\">title</th>\n"
                    + "<th scope=\"col\">description</th>\n"
                    + "<th scope=\"col\">keywords</th>\n"
                    + "<th scope=\"col\">author</th>\n"
                    + "<th scope=\"col\">creation_date</th>\n"
                    + "<th scope=\"col\">storage_date</th>\n"
                    + "<th scope=\"col\">filename</th>\n"
                    + "</tr>\n"
                    + "</thead>\n"
                    + "<tbody>\n";
            for (Image i : resultados) {
                a += "<tr>\n"
                        + PrintImageData(i)
                        + "</tr>\n";
            }
            a += "</tbody>\n"
                    + "</table>\n"
                    + "</body>\n"
                    + "</html>\n";


        } else {
            a = error("3");
        }

    } catch (SQLException ex) {
        Logger.getLogger(GenericResource.class.getName()).log(Level.SEVERE, null, ex);

    }
    return a;
}
```

Por último, para el desarrollo de la práctica 5 hemos implementado la descarga de imágenes en el servicio SOAP.

En el servicio REST no es posible descargar las imágenes por diversas razones, entre ellas que no podemos controlar el estado de la sesión. El otro motivo es por los pasos que se tienen que dar para descargar una imagen, ya que se tiene que ir al servidor, coger el contenido y llevarlo de vuelta al cliente y este último paso el que no se puede hacer.