

## II. Desarrollo de servicios web con servlets y páginas JSP

He realizado los ejercicios 2, 3 y 4 de esta parte, así que explicare el ejercicio 4 y viendo las diferencias con el resto de apartados.

Primero explicare el funcionamiento de cada una de las partes del programa y luego enseñare su funcionamiento.

El funcionamiento a rasgos generales es crear Servlets, los cuales se van a encargar de crear la sesión y redireccionar esta a través de las distintas paginas. Las paginas se crearan a través de archivos JSP, estos archivos contienen código html normal que se puede ampliar importando librerías, las cuales añaden tanto etiquetas html como nuevas funcionalidades, además de poder añadir clases de java para usarlos en los scriptlet. Estos scriptlet son etiquetas con la forma de “<% %>” donde se puede añadir código java.

Primero voy a explicar el servlet inicial, el cual se carga al poner la url base la web.

```
//Servlet inicial
@WebServlet(name = "MainPage", value = "")
public class MainPageServlet extends HttpServlet{

    public void init() {

    }

}
```

Con @WebServlet, le damos nombre al servlet y con value le decimos que url queremos para acceder a ella.

Con init() simplemente iniciamos el servlet.

```
@Override
public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException{
    //Cargamos la sesion
    String loginName = LoginManager.getLoginName(request);
    //Si hay sesion iniciada, vamos a index
    if(loginName!=null){
        request.setAttribute( "loginName", loginName);
        request.getRequestDispatcher( "index.jsp?pag=1&titulo=&autor=").forward(request, response);
    }
    //Si no hay sesion iniciada, vamos a login
    else{
        request.getRequestDispatcher( "login.jsp").forward(request, response);
    }
}
```

Con este método get, al llamar al servlet con la url, nos devolverá algo. En este caso, comprobaremos si el usuario ha iniciado sesión. En caso de que sea así, redirigimos el usuario a la pagina inicial index.

En cualquier otro caso, enviamos el usuario a la pagina de inicio de sesión. Desde la pagina de inicio de sesión llamamos al servlet de login, así que vamos a explicar primero la pagina.

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
```

El principio es como cualquier pagina escrita en html, donde lo único que cambia es la primera linea, donde podemos importar distintas librerías, en este caso importamos el charset y encoding.

Tras esto, en head, escribimos el titulo de la pagina

```
<body>
    <p>Inicia sesion para acceder a la web:</p>
    <!-- Formulario para iniciar sesion -->
    <form action="${pageContext.request.contextPath}/login" method="post">
        <label for="username">Usuario: </label><br><br>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Contraseña: </label><br><br>
        <input type="password" id="password" name="password" required><br><br>
        <input type="submit" value="Iniciar sesion"><br>
    </form>
```

En body colocamos todo lo que vamos a mostrar en la web. En este caso usamos la etiqueta <p> para mostrar un párrafo y <form> para un formulario.

Donde ponemos en action a donde vamos a enviar los datos del formulario, en este caso, ponemos la ruta al servlet de login y como method ponemos post ya que queremos enviar información.

Dentro del form tenemos varios elementos, label es una etiqueta para un campo, input sirve para escribir la información que queremos enviar. Dentro de input con type podemos indicar que tipo de información tiene que haber, con text indicamos que es un campo de texto plano y con password que es texto plano pero al escribir no se vea lo escrito. Por ultimo, con type submit creamos un botón, el cual al pulsar hará que se envíe la información del form.

```

<!-- Si el servlet devuelve error, lo mostramos -->
<%
    int error = 0;
    try{
        error = Integer.parseInt(request.getAttribute("error").toString());
    }catch(NumberFormatException e){

    }catch (NullPointerException e){}
    switch(error){
        case 1:%>
            <p style="...">Nombre de usuario o password incorrecto</p>
            <%break;
        case 2:%>
            <p style="...">Nombre de usuario o password demasiado corto</p>
            <%break;
        default:}%>

```

Tras enviar el form, pueden pasar dos cosas como veremos un poco mas adelante, que cree la sesión o que nos devuelva un error. Para este caso tengo dos tipos de errores, que el inicio de sesión sea erróneo o que los datos enviados sean demasiado cortos. Esto lo conseguimos poniendo en la sesión un atributo llamado error, el cual tendrá un valor asociado que conseguimos como se ve arriba. Y dependiendo de este, con un switch mostramos una cosa u otra.

Por ultimo, en la pagina de login tenemos:

```

<hr>
<!-- Enlace para registrarse -->
<a href="${pageContext.request.contextPath}/register">Si no tienes cuenta regístrate</a><br><br>
<!-- Enlace para recuperar contraseña -->
<a href="${pageContext.request.contextPath}/recuperarPassword">Recuperar password</a><br><br>
<!-- Enlace para acceder a la pagina sin iniciar sesion -->
<a href="index.jsp?pag=1&titulo=&autor=">Accede sin cuenta</a>

```

Los cuales son enlaces a otras paginas, como la de registro, recuperar contraseña o de entrar sin iniciar sesión.

Ahora vamos a ver el servlet:

```

@WebServlet(name="login", value = "/login")
public class ProcessLogin extends HttpServlet {
    //Metodo post

```

Primero tenemos que poner nombre y valor para la url. Tras esto creamos los métodos:

```
//Metodo post
public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException{
    //Cargamos los valores del form
    String loginName = request.getParameter("username").trim();
    String password = request.getParameter("password").trim();
    //Comprobamos si estos son lo suficientemente largos
    //En caso contrario devolvemos error
    if(loginName.length() < 3 || password.length() < 3){
        request.setAttribute("error", "2");
        request.getRequestDispatcher("login.jsp").forward(request, response);
    }
    //Iniciamos sesion
    }else if(FileUsers.login(loginName, password)){
        LoginManager.login(request, loginName.trim(), password);
        response.sendRedirect(request.getContextPath()+"/");
    }
    //Si hay algun error al iniciar sesion, devolvemos error
    }else{
        request.setAttribute("error", "1");
        request.getRequestDispatcher("login.jsp").forward(request, response);
    }
}
```

Para login solo tendremos un método Post, el cual recibirá datos desde el formulario de la pagina de login.

En este método lo primero que hacemos es conseguir los valores del formulario. Tras esto comprobamos si estos valores son validos mirando si son lo suficientemente largos. En caso de no cumplir esto devolvemos un error dándole un valor al atributo error y redireccionando de vuelta a la pagina de login. Si el inicio de sesión es correcto, creamos la sesión con el nombre usuario y volviendo a la pagina principal. Si este es incorrecto, devolvemos otro error distinto el cual significa que las credenciales son incorrectas.

Como se puede ver en la captura usamos una clase llamada loginManager, esta hay varios métodos relacionados con el inicio de sesión y la sesión.

```
//Clase que se encarga de lo relacionado con el login
public final class LoginManager{
    private final static String LOGIN_NAME_ATTRIBUTE = "loginName";

    private LoginManager(){}
```

```
//Metodo para crear sesion
public final static void login(HttpServletRequest request, String loginName, String password){
    HttpSession session = request.getSession(true);
    session.setAttribute(LOGIN_NAME_ATTRIBUTE, loginName);
}
```

Primero tenemos un método que crea un atributo con el nombre del usuario y crea la sesión.

```
//Metodo para eliminar la sesion
public final static void logout(HttpServletRequest request){
    HttpSession session = request.getSession( b: false);
    if(session!=null){
        session.invalidate();
    }
}
```

Cuando queremos cerrar sesión, llamamos a este método que borra la sesión.

```
//Metodo para devolver el nombre de usuario en la sesion
public final static String getLoginName(HttpServletRequest request){
    HttpSession session = request.getSession( b: false);
    if(session==null){
        return null;
    }else{
        return (String) session.getAttribute(LOGIN_NAME_ATTRIBUTE);
    }
}
```

Por ultimo, un método que devuelve el nombre del usuario.

También, para controlar los usuario, los cuales están en un archivo con la siguiente forma:

```
admin admin
<usuario> <password>
```



Y para leer estos datos cree una clase llamada FileUsers donde hay varios métodos para interactuar con el archivo.

```
public static boolean searchUser(String username){
    try {
        //Cargamos el archivo
        File f = new File(path);
        BufferedReader b = new BufferedReader(new FileReader(f));
        String line = "";
        //Leemos cada linea y si el usuario es igual al que buscamos, devolvemos True
        while((line = b.readLine())!=null){
            String[] partes = line.split(regex: " ");
            if(partes[0].equals(username)){
                b.close();
                return true;
            }
        }
        b.close();
        return false;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

Primero tenemos este método, el cual carga el archivo, itera sobre cada una de las líneas y comprueba si el usuario que buscamos este en el archivo.

```
//Añadimos un usuario al archivo
public static boolean addUser(String username, String password){
    //Primero comprobamos si el usuario ya existe
    if(!searchUser(username)){
        //Cargamos el archivo para escritura
        try {
            BufferedWriter bw = new BufferedWriter(new FileWriter(path, append: true));
            //Añadimos un usuario al final del fichero
            bw.newLine();
            bw.write(str: username+" "+password);
            bw.close();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        return true;
    }
    return false;
}
```

Después tenemos un método para añadir un usuario. En este primero comprobamos si el usuario ya existe, en caso de que no, abrimos el archivo, creamos una nueva

línea y añadimos el usuario junto a su contraseña. Esto nos devolvería true, para saber que se ha creado sin problemas o false si falla algo.

```
public static boolean login(String username, String password){
    //Primero vemos si el usuario existe
    if(searchUser(username)){
        try {
            //Cargamos el archivo
            File f = new File(path);
            BufferedReader b = new BufferedReader(new FileReader(f));
            String line = "";
            //Leemos el archivo hasta que veamos si existe algun usuario y contraseña que coincida
            while((line = b.readLine())!=null){
                String[] partes = line.split( regex: " ");
                if(partes[0].equals(username) && partes[1].equals(password)){
                    b.close();
                    return true;
                }
            }
            b.close();
            return false;
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
    return false;
}
```

El siguiente que tenemos es un método parecido al de buscar usuario, pero en este caso nos devuelve true si la contraseña y usuario coinciden con alguna de algún usuario.

```

public static String getPassword(String username){
    try {
        //Cargamos fichero
        File f = new File(path);
        BufferedReader b = new BufferedReader(new FileReader(f));
        String line = "";
        //Leemos cada linea, si encontramos el usuario, devolvemos la contraseña
        while((line = b.readLine())!=null){
            String[] partes = line.split(regex: " ");
            if(partes[0].equals(username)){
                b.close();
                return partes[1];
            }
        }
        b.close();
        return null;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
}

```

Por ultimo, tenemos este método, que busca un usuario y devuelve su contraseña si este existe o null en caso contrario.

Tras explicar los métodos usados, voy a seguir con la pagina de registro y su servlet. La pagina de registro es igual a la de login, la única diferencia es el action del formulario

```

<form action="${pageContext.request.contextPath}/register" method="post">

```

Que simplemente seria algo así.

Su servlet es un poco distinto, empezamos como cualquier servlet

```

@WebServlet(name="register", value="/register")
public class ProcessRegister extends HttpServlet {

```



Tras esto, tenemos un método get

```
//Metodo get
public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
    //Vemos si hay una sesion activa
    String loginName = LoginManager.getLoginName(request);
    //Si la sesion esta activa, cargamos index
    if(loginName!=null){
        request.setAttribute( s: "loginName", loginName);
        request.getRequestDispatcher( s: "index.jsp").forward(request, response);
    }else{
        //En caso contrario, cargamos la pagina de registro
        request.getRequestDispatcher( s: "register.jsp").forward(request, response);
    }
}
```

El cual comprueba si ya se ha iniciado sesión, en caso de que sea así, se redirige a la pagina index, sino a la de registro.

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
    //Cargamos los valores del formulario
    String loginName = request.getParameter( s: "username").trim();
    String password = request.getParameter( s: "password").trim();
    //Comprobamos si estos son lo suficientemente largos
    //En caso contrario, devolvemos error
    if(loginName.length()<3 || password.length()<3){
        request.setAttribute( s: "error", o: "2");
        request.getRequestDispatcher( s: "register.jsp").forward(request, response);
    }else if(FileUsers.addUser(loginName, password)){
        //Si añadimos el usuario correctamente, redigirimos a la pagina de inicio
        response.sendRedirect( s: request.getContextPath()+"/");
    }else{
        //Si sucede algun error, devolvemos error
        request.setAttribute( s: "error", o: "1");
        request.getRequestDispatcher( s: "register.jsp").forward(request, response);
    }
}
```

Por ultimo tenemos un método post, el cual recibe la información del formulario. Este funciona igual que el usado en login, pero en este caso en vez de comprobar las credenciales, creamos el usuario.

Antes de pasar a la pagina principal, tenemos la pagina de recuperar contraseña, su head es igual que al resto cambiando solo el titulo y lo importante es el body.

```
<!-- Formulario para buscar contraseña -->
<form action="${pageContext.request.contextPath}/recuperarPassword" method="post">
    <label for="username">Usuario:</label>
    <input type="text" name="username" id="username">
    <input type="submit">
</form>
```

Primero tenemos un formulario con el que enviamos el nombre del usuario al servlet de recuperar contraseña.

```
<%
    if(request.getAttribute("password")!=null){%>
        <p>Contraseña: <%=request.getAttribute("password")%></p>
    <%>else if(request.getAttribute("error")!=null){%>
        <p style="...">El usuario no existe</p>
    <%>%>
```

Tras esto tenemos un scriptlet, el cual comprueba si el atributo password se ha creado. Eso significaría que ya hemos buscado un usuario y el servlet nos ha devuelto la contraseña, así que mostraríamos esta. Por ultimo, si se ha llamado al servlet y ha devuelto error, el atributo error no sería nulo y mostramos el mensaje de error.

```
<!-- Enlace para iniciar sesion -->
<a href="{pageContext.request.contextPath}">Si tienes cuenta, haz login aqui</a><br><br>
<!-- Enlace para registrarse -->
<a href="{pageContext.request.contextPath}/register">Si no tienes cuenta registrate</a><br><br>
<!-- Enlace para acceder sin iniciar sesion -->
<a href="index.jsp?pag=1&titulo=&autor=">Accede sin cuenta</a>
```

Lo ultimo que tenemos son enlaces a las otras paginas.

Para construir la pagina index he usado un par de clases de apoyo para leer el archivo de los libros.

Primero tenemos un archivo que simplemente contiene la información de un libro

```
private String titulo, autor, enlace, resumen;
private Long descargas;

public Libros(String titulo, String autor, String enlace, String resumen, Long descargas){
    this.titulo = titulo;
    this.autor = autor;
    this.enlace = enlace;
    this.resumen = resumen;
    this.descargas = descargas;
}
```

El cual solo contiene getters y setters

Después, otra clase que lee de un archivo que tiene la siguiente forma:

```
{
  "libros": [
    {
      "libro1": {
        "titulo": "El Principito",
        "autor": "Antoine de Saint-Exupéry",
        "enlace": "/home/magui/uni/desSerTel/tema2/b",
        "resumen": "resumen1blalblalblalbal",
        "nDescargas": 0
      }
    },

```

```
public static ArrayList<Libros> busquedaLibrosjson(String titulo, String autor)
//Si ambos son vacíos devolvemos todos
if(titulo=="&&autor==""){
    return listaLibrosjson();
}
//Si el titulo esta vacío, buscamos por el autor
}else if(titulo==""){
    return listaLibrosAutorjson(autor);
}
//Si el autor esta vacío, buscamos por el titulo
}else if(autor==""){
    return listaLibrosTitulojson(titulo);
}
//Buscamos por autor y titulo
return listaLibrosAutorTitulojson(autor, titulo);
}
```

Este método es el principal de la clase que vamos a usar, le pasamos por parámetros el titulo y el autor que queremos buscar. Si ambos son vacíos, devolvemos todos los libros. Si el autor esta vacío buscamos por titulo e igual en caso contrario. Si ambos son no nulos buscamos por estos.

Cada uno de estos métodos son iguales entre ellos, la única diferencia es a la hora de que buscar. Así que voy a explicar el primero completo y luego solo las diferencias.

```
//Devuelve la lista completa de libros
public static ArrayList<Libros> listaLibrosjson() throws IOException, ParseException {
    ArrayList<Libros> libros = new ArrayList<>();
    //Cargamos el pdf y nos quedamos con los libros
    JSONParser parser = new JSONParser();
    Reader reader = new FileReader(path);
    JSONObject jsonObject = (JSONObject) parser.parse(reader);
    JSONArray array = (JSONArray) jsonObject.get("libros");
    JSONObject aux = (JSONObject) array.get(0);
    //Creamos un numero para iterar y un JSONObject para guardar los valores
    int a = 1;
    JSONObject aux2 = null;
    //Iteramos sobre el json y vamos añadiendo los libros a la lista
    while((aux2=(JSONObject) aux.get("libro"+a))!=null){
        libros.add(new Libros((String) aux2.get("titulo"),
                                (String) aux2.get("autor"), |
                                (String) aux2.get("enlace"),
                                (String) aux2.get("resumen"),
                                (Long) aux2.get("nDescargas")));
        a++;
    }
    return libros;
}
```

Este método es para devolver todos los libros. Primero creamos una lista de libros y cargamos el archivo. Por la forma del json, tenemos que cargar la key libros, y el primer objeto del array que contiene, de esta forma podemos acceder a todos los libros. Tras esto, nos creamos una variable para iterar y un json auxiliar. La forma de cualquier key en el json sera de la forma libro1, libro2, etc. Así que iteramos de esta forma hasta que no queden libros. Por cada libro, creamos un objeto libros que añadimos al array y lo devolvemos tras terminar.

En el resto de métodos solo cambia la parte de añadir al array, donde tendremos una condición para filtrar los valores que queremos.

En el caso de buscar por autor tenemos:

```
if(((String)aux2.get("autor")).contains(autor)){
    libros.add(new Libros((String) aux2.get("titu
}
}
```

En el caso de buscar por titulo tenemos:

```
if(((String)aux2.get("titulo")).contains(titulo)){
    libros.add(new Libros((String) aux2.get("titulo
}
}
```



Y por ultimo, en caso de buscar por ambos tenemos:

```
if(((String)aux2.get("autor")).contains(autor) && ((String)aux2.get("titulo")).contains(titulo)){  
    libros.add(new Libros((String) aux2.get("titulo"), (String) aux2.get("autor"), (String) aux2.get("descargas")));  
}
```

En este clase, además tenemos un método que sirve para actualizar el número de descargas de un libro. Se haría de la siguiente manera:

```
public static void updateDescargas(String titulo, String autor)  
{  
    JSONParser parser = new JSONParser();  
    Reader reader = new FileReader(path);  
    JSONObject jsonObject = (JSONObject) parser.parse(reader);  
    JSONArray array = (JSONArray) jsonObject.get("libros");  
    JSONObject aux = (JSONObject) array.get(0);  
    int a = 1;  
    JSONObject aux2 = null;  
    //Creamos un string para la salida  
    String salida = "";
```

Primero cargamos el fichero y json como antes, además creamos un String que nos servirá de salida.

```
while((aux2=(JSONObject) aux.get("libro"+a))!=null){  
    //Si el titulo y autor son los que buscamos, actualizamos el numero de descargas  
    if(((String)aux2.get("autor")).contains(autor) && ((String)aux2.get("titulo")).contains(titulo)){  
        aux2.put("nDescargas", ((Long) aux2.get("nDescargas")+1));  
    }  
    //Tras esto guardamos en el string de salida el libro actual  
    salida += "\"libro"+a+"\": "+aux2+", ";  
    a++;  
}
```

Ahora iteramos sobre la lista de libros, buscando el libro que hemos descargado para aumentar el número de descargas y vamos escribiendo todos en el string de salida.

```
//Creamos un json con el string de salida  
salida = salida.substring(0, salida.length()-1);  
salida = "{\n  \"libros\": [\n    "+salida+"  ]\n}";  
org.json.JSONObject json = new org.json.JSONObject(salida);  
  
//Escribimos el json en el archivo  
FileWriter file = new FileWriter(path);  
file.write(json.toString());  
file.flush();  
file.close();
```

Tras terminar de iterar, borramos el ultimo carácter, ya que es una coma que no queremos, y añadimos la cabecera del json. Convertimos este string a json y lo escribimos en el fichero de los libros.

Ahora toca la pagina principal, donde tenemos de primera:

```
<%@ page import="com.example.parte2.controlLibros" %>
<%@ page import="com.example.parte2.Libros" %>
<%@ page import="java.util.ArrayList" %>
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
  <head>
    <title>Index</title>
  </head>
```

Donde importamos las distintas cosas a usar y ponemos el titulo  
Lo primero que tenemos en el body es:

```
<!-- Comprobamos si esta la sesion iniciada, mostramos el usuario -->
<%if((String)session.getAttribute("loginName")!=null){%>
<pre>User: ${loginName}      <a href="${pageContext.request.contextPath}/logout">Logout</a></pre>
<hr>
<%}%>
```

Aquí vemos si el usuario ha iniciado sesión, en caso de ser así, mostramos el nombre de usuario de esa manera, y añadimos un enlace al servlet para cerrar sesión.

Después, tenemos el formulario de búsqueda:

```
<!-- Formulario para buscar libros -->
<form action="index.jsp?pag=1">
  <label for="titulo">Titulo: </label>
  <input type="text" id="titulo" name="titulo">
  <label for="autor">Autor: </label>
  <input type="text" id="autor" name="autor">
  <input type="submit" value="Buscar">
</form>
```

El cual llama de nuevo a la propia pagina index, de esta tendremos titulo y autor como parámetros en la url. Además, ponemos como parámetro que se cargue la primera pagina por defecto al buscar.



Ahora toca mostrar los libros, esto lo hacemos de la siguiente manera:

```
<table style="text-align: center">
  <!-- Headers de la tabla -->
  <tr>
    <th>Titulo</th>
    <th>Autor</th>
    <th>Enlace</th>
    <th>Resumen</th>
    <th>Descargas</th>
  </tr>
```

Primero creamos una tabla con sus cabeceras. Tras esto toca crear las filas del archivo:

```
<%
//Cargamos el numero de pagina que estamos mostrando, como default sera el 1
int pag = 1;
//En caso de que haya un parametro con numero de pagina, usamos ese
try{
    pag = Integer.parseInt(request.getParameter("pag"));
}catch(NumberFormatException e){}

//Cargamos autor y titulo desde parametros
String autor = request.getParameter("autor");
String titulo = request.getParameter("titulo");
```

Lo primero que necesitamos es ver en que pagina estamos, esta por defecto sera la primera así que creamos la variable. Después intentamos conseguir el parámetro “pag”, si este es nulo saltara error, así que lo introducimos en un try catch. De este forma, tenemos que sea 1 si no nos dicen nada o el valor que haya como parámetro. Tras esto, conseguimos de los parámetros el autor y titulo. Estos, por como hemos creado la búsqueda de libros, nos da igual que sean vacíos o tengan valores.

```
int limit = 0;
if(libros.size()>pag*5){
    limit=pag*5;
}else{
    limit=libros.size();
}
//Iteramos sobre los libros
for(int i = pag*5-5; i<limit; i++){
    Libros libro = libros.get(i);
```

Lo siguiente a hacer es el rango de los libros que vamos a mostrar, cada pagina contendrá 5 libros solo. Así que multiplicamos 5 por el numero de la pagina para el limite, aunque si el numero de libros es menor o igual a 5, nos quedamos con el tamaño de la lista de libros.

Con el for, empezamos a iterar, empezando por el numero de paginas multiplicado por 5 y restando 5, así tenemos el principio del rango.

Las celdas de todos los valores menos el enlace de descarga lo creamos de la siguiente manera:

```
<tr>
<td><%=libro.getTitulo()%></td>
<td><%=libro.getAutor()%></td>
<td><%=libro.getResumen()%></td>
<td><%=libro.getDescargas()%></td>
```

Para el enlace de descargas necesitamos un poco mas de código, ya que si el usuario no ha iniciado sesión no podrá descargar el libro.

```
<%if((String)session.getAttribute("loginName")==null){%>
<td><a href="{pageContext.request.contextPath}">Inicia sesion para poder descargar</a></td>
<%}else{%>
<!-- Mostramos el enlace de descarga como un boton de submit para un form invisible
      Cuando se pulsa, enviamos al servlet de descargar pdf el enlace, titulo y autor del libro
      El enlace para descargar el libro
      El autor y titulo para actualizar el numero de descargas-->
<td><form action="{pageContext.request.contextPath}/descargarPdf" method="get">
  <input type="hidden" name="enlace" id="enlace" value="{libro.getEnlace()}">
  <input type="hidden" name="titulo" id="titulo2" value="{libro.getTitulo()}">
  <input type="hidden" name="autor" id="autor2" value="{libro.getAutor()}">
  <input type="submit" name="boton" id="boton"
    value="{libro.getEnlace().split("\\\\")[libro.getEnlace().split("\\\\").length-1]}">
</form></td>
```

Así que tendremos ese código. Primero comprobamos que haya una sesión, en caso de que no la haya, mostramos un enlace para la pagina de login.

Si el usuario puede descargar el libro, creamos un formulario con campos escondidos, los cuales contienen el titulo, autor y enlace para poder descargar. El botón de submit se mostrara como un botón con el nombre del archivo que sacamos del final del enlace.

El servlet de descarga se vería de la siguiente manera:

```
@WebServlet(name = "Descargar", value = "/descargarPdf")
public class downloadPdf extends HttpServlet {
```

Como la cabecera de todos los servlet.

```

public void doGet(HttpServletRequest request,
                  HttpServletResponse response) throws IOException {
    //Cargamos el archivo para descargar
    File file = new File(request.getParameter( s: "enlace"));
    FileInputStream fileIn = new FileInputStream(file);
    ServletOutputStream out = response.getOutputStream();
    //Mandamos el archivo
    byte[] outputByte = new byte[4096];
    while(fileIn.read(outputByte, off: 0, len: 4096) != -1)
    {
        out.write(outputByte, off: 0, len: 4096);
    }
    //Actualizamos el numero de descargas de archivo
    try {
        controlLibros.updateDescargas(request.getParameter( s: "titulo"), request.getParameter( s: "
    } catch (ParseException e) {
        throw new RuntimeException(e);
    }
    fileIn.close();
    out.flush();
    out.close();
}

```

Y un método get, el cual abre el archivo de la ruta que hemos enviado con el formulario y lo envía al usuario con un ServletOutputStream. Además, llamamos al método de updateDescargas con el título y autor del formulario para actualizar las descargas del libro.

Continuando con index, tras mostrar los libros nos queda mostrar las páginas disponibles de la lista de la siguiente manera:

```

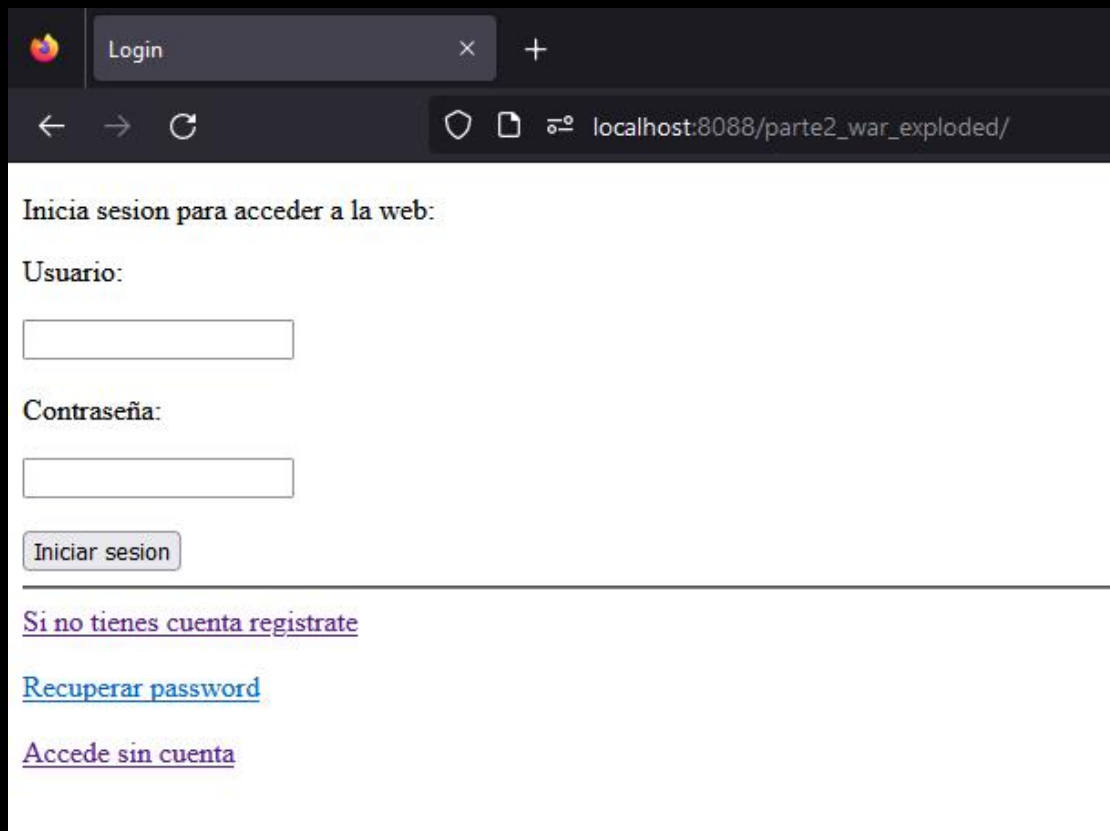
<hr>
<!-- Mostramos el numero de paginas como enlaces si hay mas de una disponible -->
<%for(int j = 1; j<=controlLibros.busquedaLibrosjson(titulo, autor).size()/5; j++){
    if(pag==j){%>
        <span><%=pag%></span>
        <%}else{%>
            <a href="index.jsp?pag=<%=j%>&titulo=<%=titulo%>&autor=<%=autor%>"><%=j%></a>
        <%}}
    %>
    <%if(controlLibros.busquedaLibrosjson(titulo, autor).size()/5>1){%>
        <hr>
        <%}%>
    }

```

Donde por cada página disponible, creamos un enlace con los parámetros de la página a la que queremos ir, el título y autor actual que estamos mirando. La página actual en la que estamos la mostramos como texto plano. En caso de que no haya páginas disponibles, no mostraríamos nada en el lugar de las páginas.

Esto sería el funcionamiento de todo del código de la práctica, ahora toca ver cómo funciona.

Si ejecutamos el proyecto y accedemos a la pagina veremos lo siguiente:



The screenshot shows a web browser window with a single tab titled 'Login'. The address bar displays 'localhost:8088/parte2\_war\_exploded/'. The page content includes a heading 'Inicia sesion para acceder a la web:', followed by a 'Usuario:' label and an empty text input field. Below this is a 'Contraseña:' label and another empty text input field. A button labeled 'Iniciar sesion' is positioned below the password field. At the bottom of the page, there are three links: 'Si no tienes cuenta regístrate' (purple), 'Recuperar password' (blue), and 'Accede sin cuenta' (purple).

Inicia sesion para acceder a la web:

Usuario:

Contraseña:

Iniciar sesion

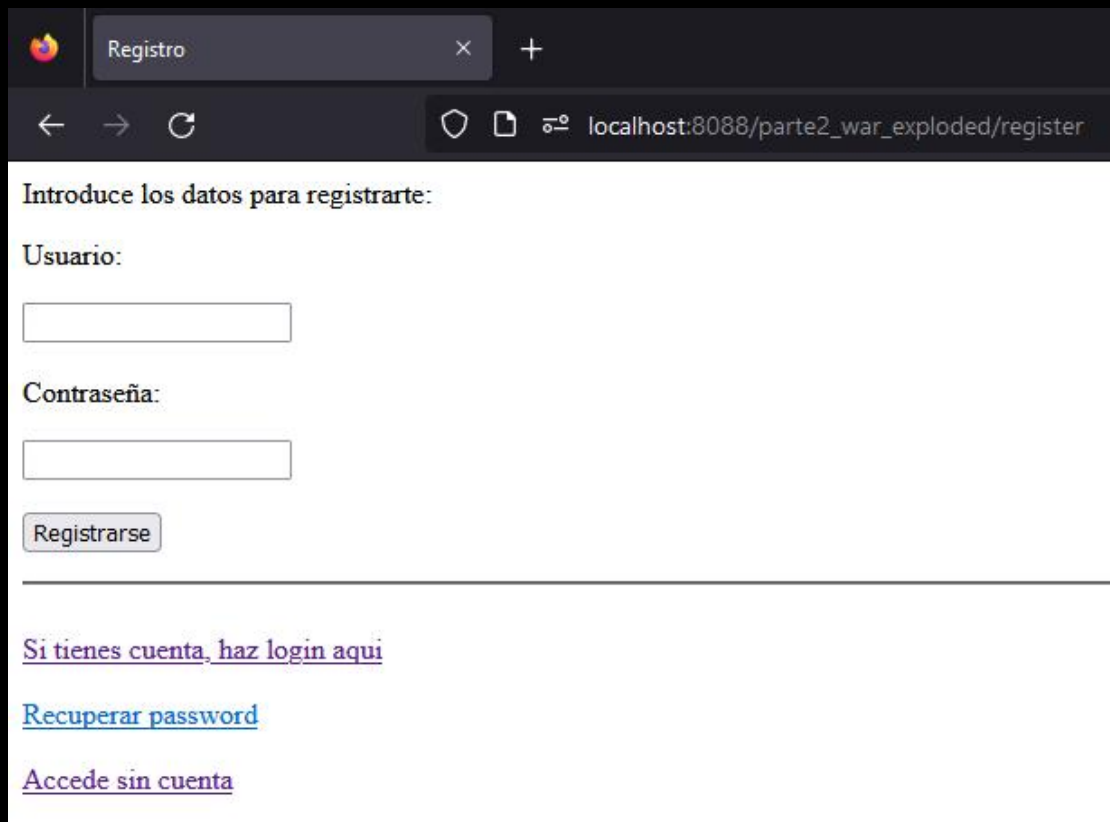
---

[Si no tienes cuenta regístrate](#)

[Recuperar password](#)

[Accede sin cuenta](#)

Veremos el formulario de inicio de sesión, como es la primera vez que entramos no tenemos un usuario creado, así que vamos a crearlo pulsando en el enlace de registro. Tras esto veremos:



The screenshot shows a web browser window with the title 'Registro'. The address bar displays 'localhost:8088/parte2\_war\_exploded/register'. The page content includes the instruction 'Introduce los datos para registrarte:' followed by two input fields labeled 'Usuario:' and 'Contraseña:'. A 'Registrarse' button is positioned below the password field. A horizontal line separates the registration form from the login links below. The links are: '[Si tienes cuenta, haz login aqui](#)', '[Recuperar password](#)', and '[Accede sin cuenta](#)'.

Registro

← → ↻ 🔒 📄 🌐 localhost:8088/parte2\_war\_exploded/register

Introduce los datos para registrarte:

Usuario:

Contraseña:

Registrarse


---

[Si tienes cuenta, haz login aqui](#)

[Recuperar password](#)

[Accede sin cuenta](#)

Vamos a introducir un usuario y contraseña para crear:



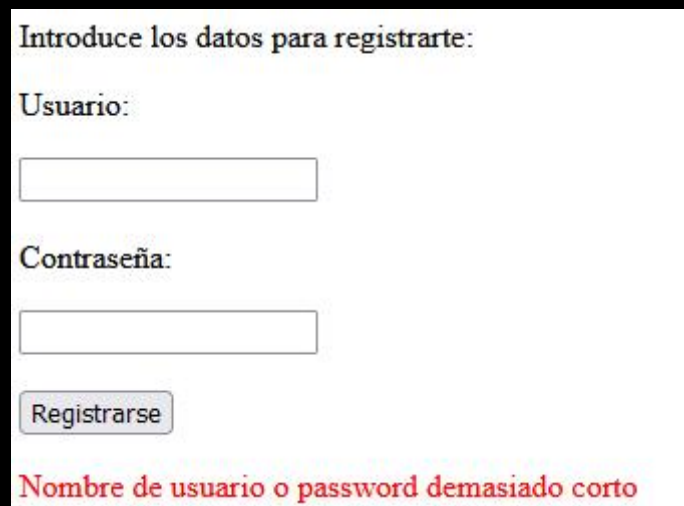
Introduce los datos para registrarte:

Usuario:

Contraseña:

Registrarse

Si pulsamos, registrarse saldrá lo siguiente:



Introduce los datos para registrarte:

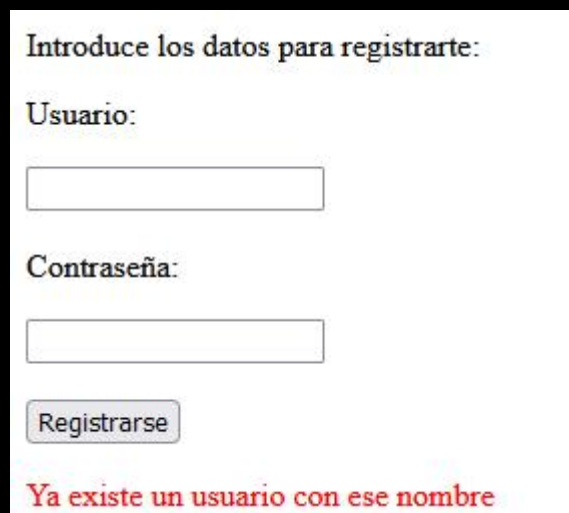
Usuario:

Contraseña:

Registrarse

**Nombre de usuario o password demasiado corto**

Esto pasa ya que la contraseña es demasiado corta. Si ponemos un usuario que ya existe como "admin" saldría lo siguiente:



Introduce los datos para registrarte:

Usuario:

Contraseña:

Registrarse

**Ya existe un usuario con ese nombre**



Vamos a crear un usuario bien:



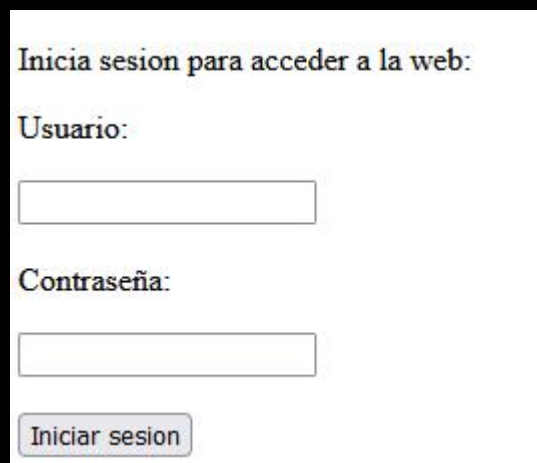
Introduce los datos para registrarte:

Usuario:

Contraseña:

Registrarse

Y tras pulsar el botón vemos:



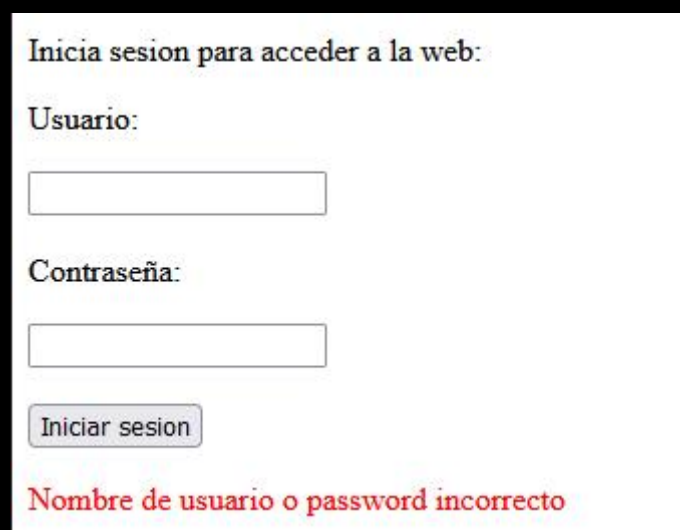
Inicia sesion para acceder a la web:

Usuario:

Contraseña:

Iniciar sesion

Nos lleva a la pagina de login de nuevo, ya que el registro ha ido bien. Vamos a iniciar sesión pero poniendo la contraseña mal:



Inicia sesion para acceder a la web:

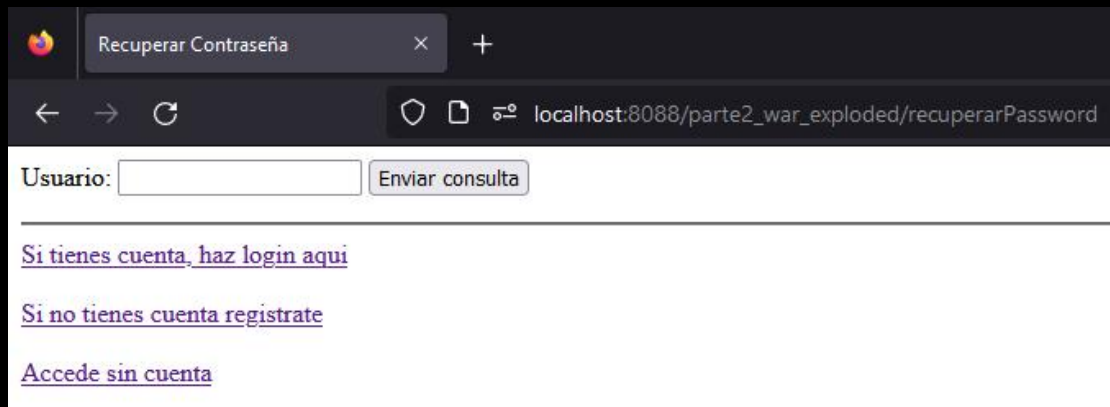
Usuario:

Contraseña:

Iniciar sesion

Nombre de usuario o password incorrecto

Y lo mismo pasaría si fuera demasiado corto. En caso de no acordarnos de la contraseña pulsaríamos el enlace para recordarla y veríamos:



Recuperar Contraseña

← → ↻ localhost:8088/parte2\_war\_exploded/recuperarPassword

Usuario:  Enviar consulta

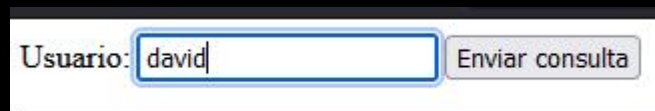
---

[Si tienes cuenta, haz login aqui](#)

[Si no tienes cuenta regístrate](#)

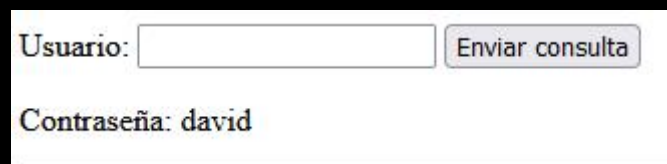
[Accede sin cuenta](#)

Donde podemos introducir el usuario para recuperar la contraseña. Así que vamos a introducir el usuario que hemos creado antes:



Usuario:  Enviar consulta

Y tras pulsar el botón vemos:



Usuario:  Enviar consulta

Contraseña: david

Si ponemos un usuario que no existe saldría:



Usuario:  Enviar consulta

El usuario no existe

Ahora vamos a iniciar sesión y ver la pagina principal:

User: david [Logout](#)

Titulo:  Autor:

Titulo	Autor	Enlace	Resumen	Descargas
El Principito	Antoine de Saint-Exupéry	<a href="#">ElPrincipito.pdf</a>	resumen1blalblalblalbal	1
Abuelita	Hans Christian Andersen	<a href="#">Abuelita.pdf</a>	resumen1blalblalblalbal	0
Alicia en el pais de las maravillas	Lewis Carroll	<a href="#">AliciaEnElPaisDeLasMaravillas.pdf</a>	resumen1blalblalblalbal	0
Barbanegra	Daniel Defoe	<a href="#">Barbanegra.pdf</a>	resumen1blalblalblalbal	0
Caperuticta Roja	Charles Perrault	<a href="#">CaperucitaRoja.pdf</a>	resumen1blalblalblalbal	0

[1](#) [2](#) [3](#)

Podemos ver los distintos libros que hay, vamos a cambiar a la pagina 2:

User: david [Logout](#)

Titulo:  Autor:

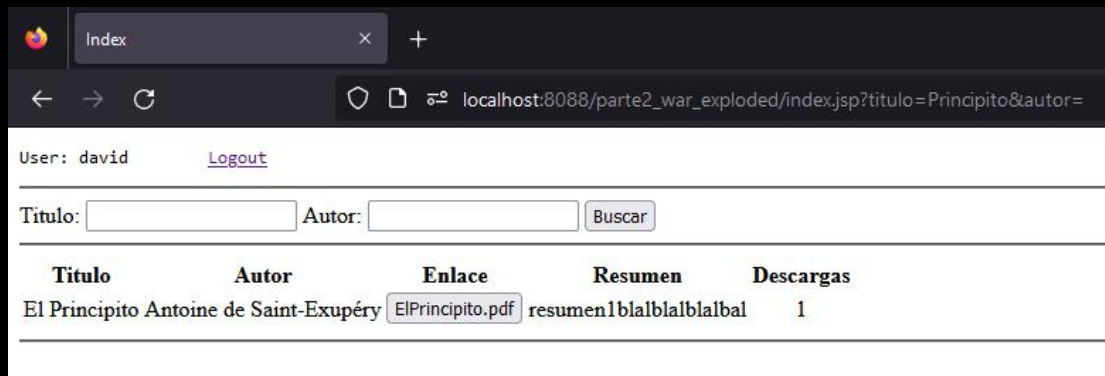
Titulo	Autor	Enlace	Resumen	Descargas
El abecedario	Hans Christian Andersen	<a href="#">ElAbecedario.pdf</a>	resumen1blalblalblalbal	0
El ave fenix	Hans Christian Andersen	<a href="#">ElAveFenix.pdf</a>	resumen1blalblalblalbal	1
El patito feo	Hans Christian Andersen	<a href="#">Elpatitofeo.pdf</a>	resumen1blalblalblalbal	0
Hansel y Gretel	Hermanos Grimm	<a href="#">HanselYGretel.pdf</a>	resumen1blalblalblalbal	0
La Bella y la Bestia	Gabrielle-Suzanne Barbot de Villeneuve	<a href="#">LaBellaYLaBestia.pdf</a>	resumen1blalblalblalbal	0

[1](#) [2](#) [3](#)

Como podemos ver, en la url están los parámetros y hemos cambiado de libros. Ahora vamos a buscar el principito:

Titulo:  Autor:

Y tras pulsar el botón vemos:



The screenshot shows a web browser window with the address bar displaying 'localhost:8088/parte2\_war\_exploded/index.jsp?titulo=Principito&autor='. The page content includes a user login section with 'User: david' and a 'Logout' link. Below this is a search form with 'Titulo:' and 'Autor:' input fields and a 'Buscar' button. The search results are displayed in a table with the following data:

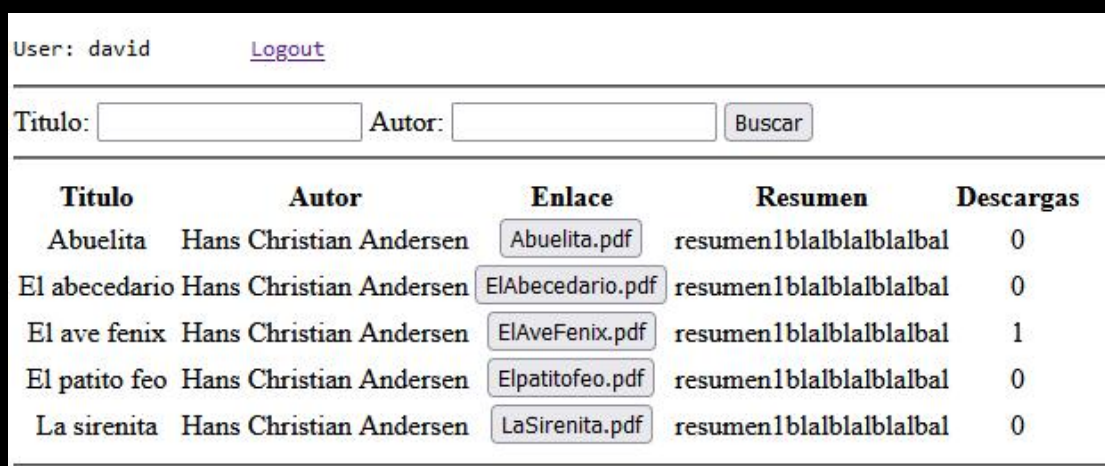
Titulo	Autor	Enlace	Resumen	Descargas
El Principito	Antoine de Saint-Exupéry	<a href="#">ElPrincipito.pdf</a>	resumen1blalblalblalbal	1

También podemos buscar por autor, por ejemplo Hans:



The screenshot shows the search form with 'Titulo:' and 'Autor:' input fields and a 'Buscar' button. The 'Autor' field contains the text 'Hans'.

Y tras pulsar el botón vemos:



The screenshot shows the web browser window with the search results for 'Hans'. The page content includes the user login section and the search form. The search results are displayed in a table with the following data:

Titulo	Autor	Enlace	Resumen	Descargas
Abuelita	Hans Christian Andersen	<a href="#">Abuelita.pdf</a>	resumen1blalblalblalbal	0
El abecedario	Hans Christian Andersen	<a href="#">ElAbecedario.pdf</a>	resumen1blalblalblalbal	0
El ave fenix	Hans Christian Andersen	<a href="#">ElAveFenix.pdf</a>	resumen1blalblalblalbal	1
El patito feo	Hans Christian Andersen	<a href="#">Elpatitofeo.pdf</a>	resumen1blalblalblalbal	0
La sirenita	Hans Christian Andersen	<a href="#">LaSirenita.pdf</a>	resumen1blalblalblalbal	0

Por ultimo podemos buscar autor y titulo a la vez, por ejemplo:

Titulo:  Autor:

Y veríamos:

Firefox View

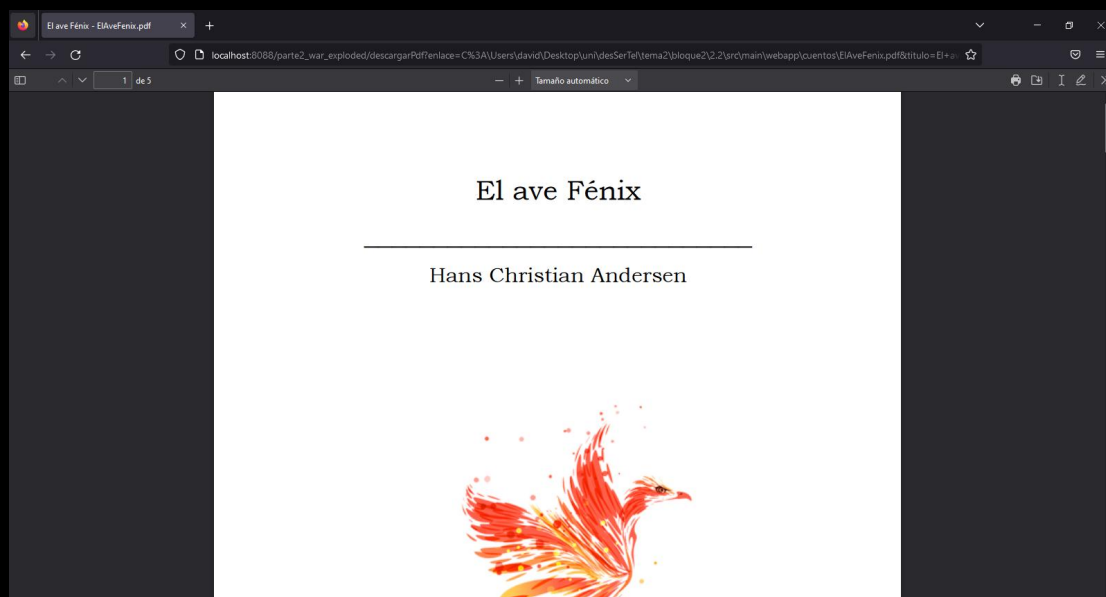
localhost:8088/parte2\_war\_exploded/index.jsp?titulo=ave&autor=Hans

User: david [Logout](#)

Titulo:  Autor:

Titulo	Autor	Enlace	Resumen	Descargas
El ave fenix	Hans Christian Andersen	<a href="#">ElAveFenix.pdf</a>	resumen1blalblalblalbal	1

Si pulsamos el botón de descarga de cualquier libro, veríamos lo siguiente:



Y si volvemos a la pagina de antes, veríamos lo siguiente:

User: david [Logout](#)

Titulo:  Autor:

Titulo	Autor	Enlace	Resumen	Descargas
El ave fenix	Hans Christian Andersen	<a href="#">ElAveFenix.pdf</a>	resumen1 blalblalblal	2

Como se puede apreciar, el numero de descargas a aumentando.

Nos queda por ver el botón de logout, el cual si pulsamos nos cierra sesión y nos lleva a la pagina de login:

Inicia sesion para acceder a la web:

Usuario:

Contraseña:

[Si no tienes cuenta regístrate](#)

[Recuperar password](#)

[Accede sin cuenta](#)



Y si accedemos a la web anterior sin sesión con el enlace vemos lo siguiente:



Titulo	Autor	Enlace	Resumen	Descargas
El Principito	Antoine de Saint-Exupéry	<a href="#">Inicia sesion para poder descargar</a>	resumen1blalblalblal	1
Abuelita	Hans Christian Andersen	<a href="#">Inicia sesion para poder descargar</a>	resumen1blalblalblal	0
Alicia en el pais de las maravillas	Lewis Carroll	<a href="#">Inicia sesion para poder descargar</a>	resumen1blalblalblal	0
Barbanegra	Daniel Defoe	<a href="#">Inicia sesion para poder descargar</a>	resumen1blalblalblal	0
Caperuticta Roja	Charles Perrault	<a href="#">Inicia sesion para poder descargar</a>	resumen1blalblalblal	0

1 2 3

Tenemos acceso completo a ver los libros, pero no podemos descargarlos, ya que necesitamos iniciar sesión y no aparecen el nombre del usuario ya que no hay.

Eso seria todo sobre el proyecto en funcionamiento.

Para toda la practica he usado JSP en vez de solo servlet, lo cual hace que la practica sea distinta a hacer solo la parte obligatorio.

Usando JSP, toda la creación de contenido se hace en las paginas JSP en vez de en los servlet.

Es decir, si usáramos solo servlets, deberíamos tener métodos get, los cuales al llamarlos, usando prints, crearíamos las paginas. En estos prints, escribiríamos el contenido de la pagina como lo haríamos de normal, usando etiquetas html pero en string. Ademas, no podríamos usar las etiquetas de scriptlet, así que todo ese contenido lo tendríamos que pasar a un string y añadirlo al print.

Ademas, al redirigir entre las distintas paginas, lo haríamos poniendo las url de los servlet en vez de poner el nombre de los archivos jsp.

De esta manera, aplicamos el modelo MVC, el cual define la separación de la interfaz de usuario, lógica y datos en distintos componentes. En este caso, la lógica serian los servlet, ya que se encargan de ver si los formularios están bien, los datos los guardamos en archivos txt o json y la interfaz de usuario que se encargan los jsp. Lo cual nos permite diferenciar mejor que tenemos que hacer en cada parte y donde pueden estar los errores.