

DESPLIEGUE APLICACIONES WEB (DAW)



CURSO 2017-2018

11 / 03 / 2018

PRÁCTICA TEMA 4

JAVA EE

Por:
José Luis Ferrete Olarte

Licencia Creative Commons BY-NC-SA

Los contenidos de este documento se publican bajo licencia [CC BY-NC-SA](#): Creative Commons Reconocimiento – NoComercial – CompartirIgual.

**Cedemos los siguientes derechos sobre la obra:**

Derecho de reproducción, distribución y comunicación pública sobre la obra.

Derecho a incorporarla en una o más obras conjuntas o bases de datos y para su reproducción en tanto que incorporada a dichas obras conjuntas o bases de datos.

Derecho para efectuar cualquier transformación sobre la obra y crear y reproducir obras derivadas.

Derecho de distribución y comunicación pública de copias o grabaciones de la obra, como incorporada a obras conjuntas o bases de datos.

Derecho de distribución y comunicación pública de copias o grabaciones de la obra, por medio de una obra derivada.

Siempre que lo hagas con estas condiciones:

Reconocer la autoría, especificando la firma y el autor que lo publica (*"D. José Luis Ferrete Olarte"*). Si es en formato digital, debes añadir un enlace al contenido original.

Compartir bajo la misma licencia. Si reproduces o remezclas esta obra, sólo puedes distribuir la obra generada bajo una licencia como ésta (con las mismas condiciones). Hemos realizado este trabajo con fines educativos y queremos que, si nos utilizas como base para tus creaciones, también contribuyas a la comunidad difundiéndolas con licencias libres. Así, aprenderemos todos juntos.

No se permite un uso comercial de la obra original, ni de las posibles obras derivadas.

Ante cualquier duda sobre las condiciones de cesión de derechos, consultar: <https://creativecommons.org/licenses/by-nc-sa/2.5/es/legalcode.es>

INDICE

| | | |
|-------|--|----|
| I. | Introducción | 3 |
| II. | Pasos previos..... | 4 |
| III. | Base de datos | 7 |
| A. | Diagrama Entidad-Relación en DIA (ERD) | 7 |
| B. | Esquema relacional (ER)..... | 8 |
| C. | Diagrama en Workbench | 8 |
| D. | Generando las sentencias SQL | 8 |
| IV. | Entrada logueada | 12 |
| V. | Aplicación Web..... | 19 |
| A. | Listar Autores | 21 |
| B. | Listar Editoriales | 28 |
| C. | Listar pedidos y modificar fecha de envío..... | 32 |
| D. | Listado de libros | 42 |
| E. | Insertar un nuevo libro..... | 52 |
| F. | Modificar libro..... | 56 |
| VI. | Despliegue .WAR | 61 |
| VII. | Apéndice..... | 66 |
| A. | Recuperar pass SUDO..... | 66 |
| VIII. | Bibliografía | 70 |

I. Introducción

En esta práctica vamos a acometer la construcción de una aplicación web de gestión para una tienda de libros online. Básicamente, debemos generar una aplicación CRUD, con su propio pool de conexiones a una base de datos de MySQL.

Nuestro proyecto se cimentará en el MVC (Modelo Vista Controlador), por ello, deberemos crear una serie de Modelos (Clases de Java), Vistas (páginas Jsp) y Controladores (Servlets de Java).

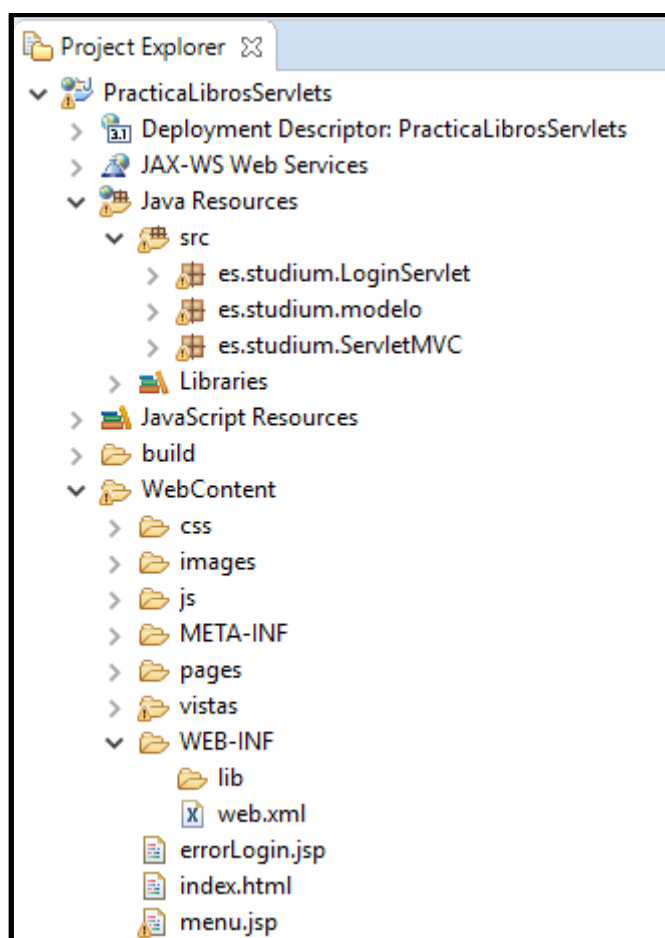
Para ello, deberemos de realizar una serie de subtarefas que iremos desgranando y comentando a lo largo de la presente práctica.

II. Pasos previos

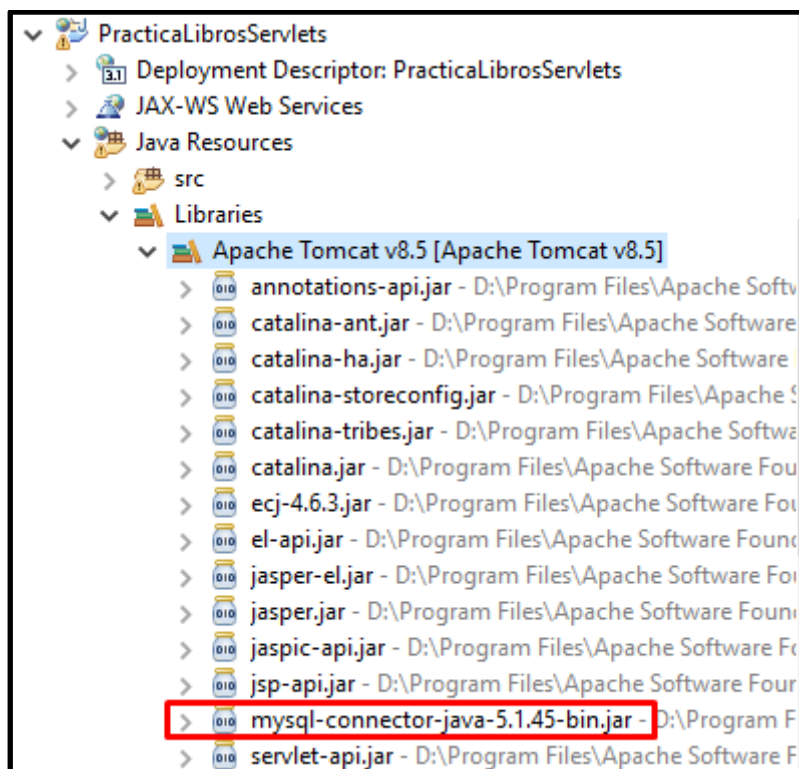
Antes de poder acometer la obra que se nos ha encomendado, debemos tener en cuenta que hay una serie de configuraciones que tenemos que realizar.

Si desplegamos el proyecto desde el explorador de proyectos, en perspectiva JavaEE, podemos ver la estructura por defecto que crea Eclipse para los proyectos Web de la plataforma Java EE:

- En la carpeta Java Resources: src guardaremos el código fuente de las clases Java empaquetadas en packages.
- En la carpeta WebContent guardaremos los archivos Web (HTML, JavaScript, CSS, JSP, imágenes, documentos, etc).
- En la carpeta WebContent/WEB-INF guardaremos el descriptor de despliegue web.xml.
- En la carpeta WebContent/WEB-INF/lib guardaremos las librerías externas que utilicemos en la aplicación Web Java.
- En la carpeta Java Resources: src, dentro de Libraries, podemos encontrar las librerías de Apache Tomcat, donde se encuentra las librerías necesarias para desarrollar Servlets, en concreto servlet-api.jar.

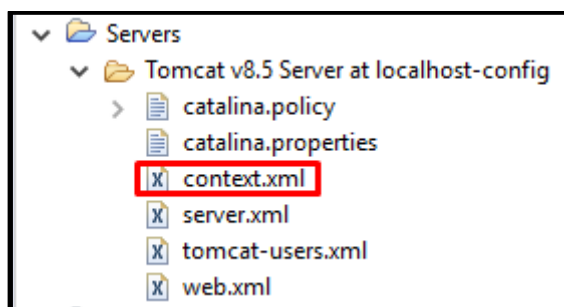


Pero antes, debemos colocar el driver JDBC, el fichero mysql-connector-java-X.Y.Z-bin.jar, en el directorio CATALINA_HOME/lib.



Sin este conector, cualquier intento de acceso a la base de datos resultará fútil.

Adicionalmente, debemos cerciorarnos de que el contexto está correctamente configurado. Para ello, navegaremos fuera de nuestro proyecto y buscaremos "context.xml" en la localización que se indica en la captura siguiente:



El fichero debería incluir lo siguiente, para garantizar la correcta configuración del recurso JNDI DataSource. De esta forma gestionamos el correcto funcionamiento de nuestro pool de conexiones y la correcta conexión a la base de datos:

```
12<Context>
13
14    <!-- Default set of monitored resources. If one of these changes, the -->
15    <!-- web application will be reloaded. -->
16    <WatchedResource>WEB-INF/web.xml</WatchedResource>
17    <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>
18
19    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
20    <!-- <Manager pathname="" /> -->
21    <Resource name="jdbc/mysql_tiendalibros" auth="Container"
22        type="javax.sql.DataSource" maxActive="100" maxIdle="30" maxWait="10000"
23        removeAbandoned="true" username="servletUser" password="Stadium2017;"
24        driverClassName="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost:3306/tiendalibros" />
25 </Context>
```

Realizados estos pasos previos de configuración JAVA EE, podemos acometer la remodelación de la base de datos.

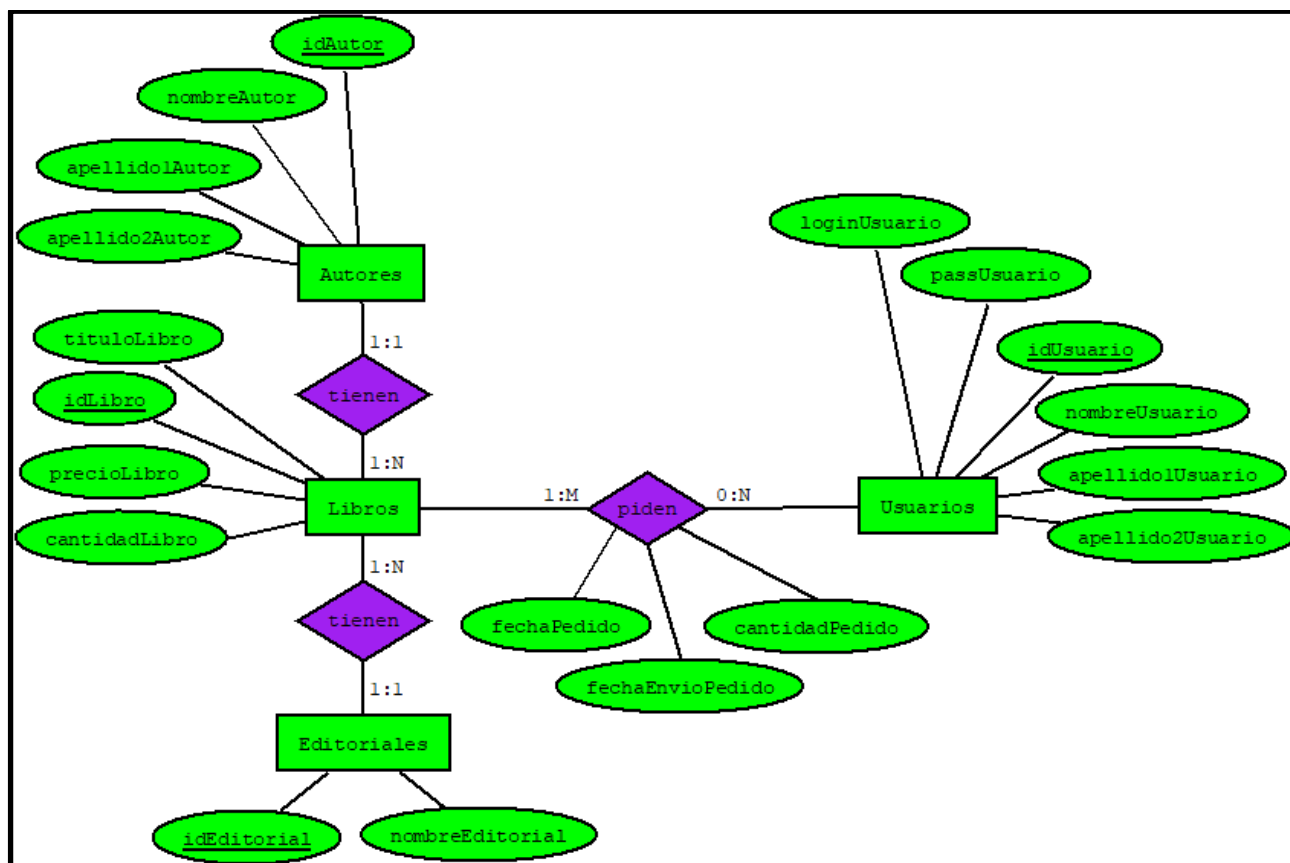
III. Base de datos

En primer lugar, lo que debemos hacer es completar y mejorar la base de datos existente. Tenemos que transformar la tabla libros que tenemos en libros, autores y editoriales. También necesitaremos la tabla de usuarios y una tabla de pedidos.

En nuestro caso concreto, hemos decidido acometer una reforma completa. Sólo hemos mantenido la base de datos (con los usuarios y las pertinentes conexiones, así como el acceso remoto desde MySQL). Todas las tablas han sido eliminadas de la base de datos. En este capítulo, analizaremos paso a paso el procedimiento empleado para la reforma integral.

A. Diagrama Entidad-Relación en DIA (ERD)

Se han tenido en cuenta las entidades y atributos requeridas en la documentación, así como otras que, aunque no se especificaban, creemos que son necesarias. Por ejemplo, para un correcto funcionamiento del pool de conexiones, necesitaremos un "loginUsuario" y un "passUsuario".



B. Esquema relacional (ER)

usuarios (*idUsuario, nombreUsuario, apellido1Usuario, apellido2Usuario, loginUsuario, passUsuario)

autores (*idAutor, nombreAutor, apellido1Autor, apellido2Autor)

editoriales (*idEditorial, nombreEditorial)

libros (*idLibro, tituloLibro, precioLibro, cantidadLibro, idEditorialFK, idAutorFK)

pedidos (*idPedido, fechaEnvioPedido, fechaPedido, cantidadPedido, idUsuarioFK, idLibroFK)

C. Diagrama en Workbench



D. Generando las sentencias SQL

El siguiente paso consiste en generar las sentencias SQL que emplearemos para dar forma a nuestra base de datos. Son las siguientes:

```
CREATE DATABASE tiendalibros CHARACTER SET utf8 COLLATE utf8_spanish2_ci;
```

```
use tiendalibros;
```

```
CREATE TABLE usuarios (  
idUsuario INT NOT NULL AUTO_INCREMENT UNIQUE,  
nombreUsuario VARCHAR(45) NOT NULL,  
apellido1Usuario VARCHAR(45) NOT NULL,  
apellido2Usuario VARCHAR(45) NOT NULL,  
loginUsuario VARCHAR(45) NOT NULL,  
passUsuario VARCHAR(256) NOT NULL,
```

```
PRIMARY KEY(idUsuario)
);
```

```
CREATE TABLE autores (
idAutor INT NOT NULL AUTO_INCREMENT UNIQUE,
nombreAutor VARCHAR(45) NOT NULL,
apellido1Autor VARCHAR(45) NOT NULL,
apellido2Autor VARCHAR(45) NOT NULL,
PRIMARY KEY(idAutor)
);
```

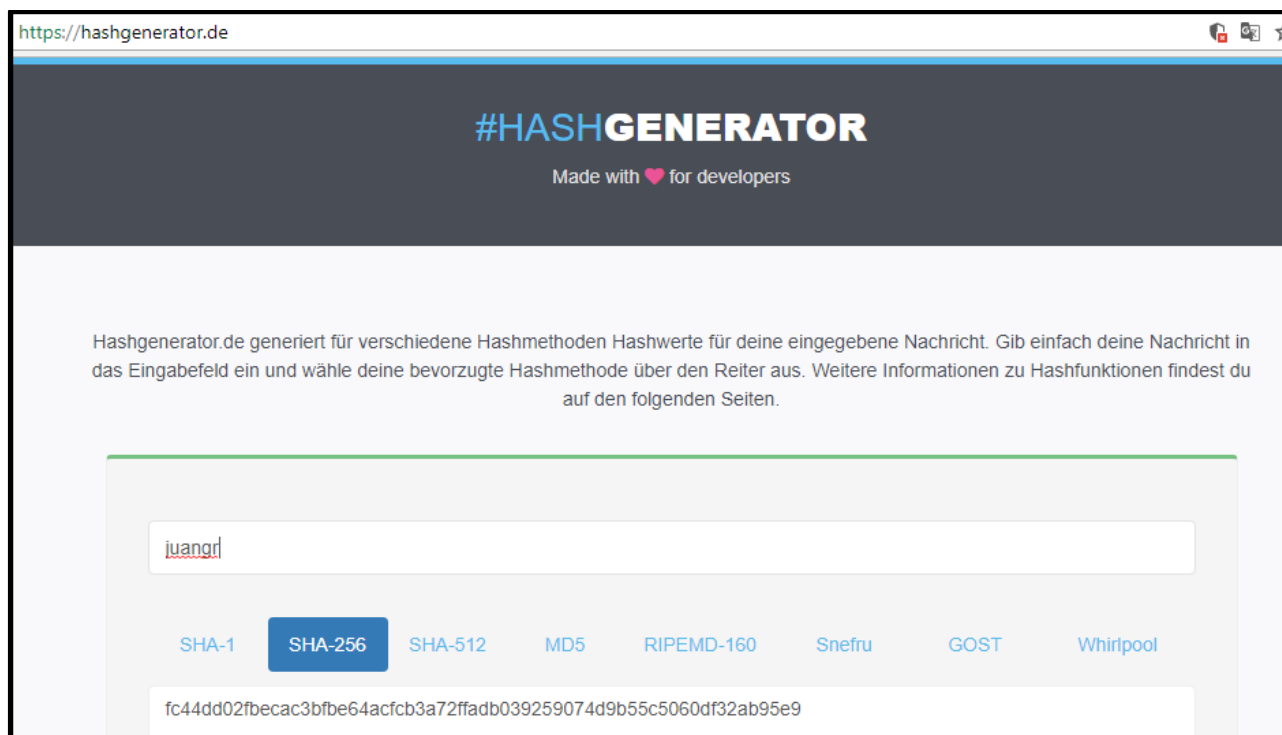
```
CREATE TABLE editoriales (
idEditorial INT NOT NULL AUTO_INCREMENT UNIQUE,
nombreEditorial VARCHAR(90) NOT NULL,
PRIMARY KEY(idEditorial)
);
```

```
CREATE TABLE libros (
idLibro INT NOT NULL AUTO_INCREMENT UNIQUE,
tituloLibro VARCHAR(45) NOT NULL,
precioLibro DECIMAL(6,2) NOT NULL,
cantidadLibro INT(4) NOT NULL,
idEditorialFK INT NOT NULL,
idAutorFK INT NOT NULL,
PRIMARY KEY(idLibro),
FOREIGN KEY (idEditorialFK) REFERENCES Editoriales(idEditorial),
FOREIGN KEY (idAutorFK) REFERENCES Autores(idAutor)
);
```

```
CREATE TABLE pedidos (
idPedido INT NOT NULL AUTO_INCREMENT UNIQUE,
fechaEnvioPedido DATE,
fechaPedido DATE NOT NULL,
cantidadPedido INT NOT NULL,
idUsuarioFK INT NOT NULL,
idLibroFK INT NOT NULL,
PRIMARY KEY(idPedido),
FOREIGN KEY (idUsuarioFK) REFERENCES Usuarios(idUsuario),
FOREIGN KEY (idLibroFK) REFERENCES Libros(idLibro)
);
```

```
INSERT INTO usuarios VALUES
(DEFAULT,"Juan","Garcia","Ramirez","juangr","fc44dd02fbecac3bfbe64acfc3a72ffad
b039259074d9b55c5060df32ab95e9"),
(DEFAULT,"Maria","Pelaez","Garcia","mariapg","c1697b1de2b5f5e50848c3a21562def56
9d3f8e824787221fcfbac5bc4b845");
```

Antes de continuar con las sentencias, debemos realizar una aclaración sobre la generación de los "passUsuario". Queríamos otorgar un plus de seguridad a nuestra aplicación web con entrada logueada, por ello, hemos recurrido a encriptar las contraseñas con SHA-256. Este factor, deberá ser tenido en cuenta posteriormente en nuestro código y evitar realizar una doble encriptación, que podría darnos más de un dolor de cabeza.



Tras este inciso, continuamos con las sentencias SQL:

```
INSERT INTO autores VALUES (DEFAULT,"George","R.R.","Martin"),  
(DEFAULT,"J","R.R.","Tolkien"),  
(DEFAULT,"Arturo","Pérez","Reverte");
```

```
INSERT INTO editoriales VALUES (DEFAULT,"Gilgamesh"),  
(DEFAULT,"Alianza Editorial"),  
(DEFAULT,"Editorial Planeta");
```

```
INSERT INTO libros VALUES (DEFAULT,"Juego de Tronos",25.99,10,1,1),  
(DEFAULT,"El Señor de los Anillos",22.99,10,2,2),  
(DEFAULT,"El Capitán Alatriste",23.99,10,3,3);
```

```
INSERT INTO pedidos VALUES (DEFAULT,NULL,"2018/01/26",1,1,1),  
(DEFAULT,NULL,"2018/01/28",1,2,3),  
(DEFAULT,NULL,"2018/01/29",1,1,2);
```

Estos valores NULL, corresponden a la fecha de envío del pedido y serán modificados a posteriori, con nuestra aplicación web.

Adicionalmente a estas instrucciones, debemos garantizar que existe el usuario que hemos definido en nuestra base de datos, para ello, realizaremos los siguientes pasos:

```
mysql> use tiendalibros;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show grants for servletUser;
ERROR 1141 (42000): There is no such grant defined for user 'servletUser' on host '%'
mysql> create user 'servletUser' IDENTIFIED BY 'Stadium2017;';
Query OK, 0 rows affected (0.73 sec)

mysql> show grants for servletUser;
+-----+
| Grants for servletUser@% |
+-----+
| GRANT USAGE ON *.* TO 'servletUser'@'%' |
+-----+
1 row in set (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON tiendaLibros.* TO 'servletUser'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql> show grants for servletUser;
+-----+
| Grants for servletUser@% |
+-----+
| GRANT USAGE ON *.* TO 'servletUser'@'%' |
| GRANT ALL PRIVILEGES ON `tiendaLibros`.* TO 'servletUser'@'%' |
+-----+
2 rows in set (0.00 sec)

mysql>
```

IV. Entrada logueada

Comenzando con nuestro proyecto, lo primero es iniciar Eclipse y creamos un proyecto nuevo tipo "Dynamic Web Project". No olvidar marcar para crear un descriptor de despliegue web.xml.

Como base para este capítulo, vamos a utilizar un ejercicio anterior de pool de conexiones visto en clase. A este proyecto, deberemos realizarle numerosas modificaciones para alcanzar nuestros objetivos, pero dispondremos de una base sólida desde la que comenzar.

En nuestro caso, hemos decidido que las vistas sean generadas en ficheros Jsp, por tanto, debemos eliminar cualquiera de estas referencias que crean contenido u obtendremos cuantiosos errores, cosa que nos sucedió.

```

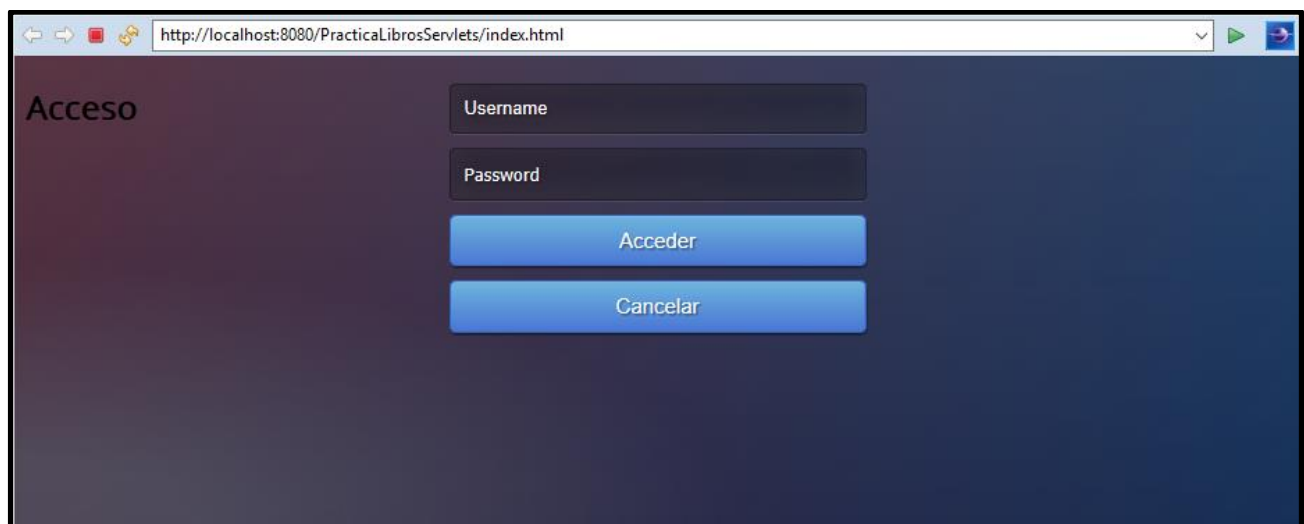
44  {
45      //response.setContentType("text/html;charset=UTF-8");
46      //PrintWriter out = response.getWriter();
47      Connection conn = null;
48      Statement stmt = null;
49      try
50      {
51          /*out.println("<html>");
52          out.println("<head>");
53          out.println("<title>Login</title>");
54          out.println("</head>");
55          out.println("<body>");
56          out.println("<h2>Login</h2>");*/
57          // Obtener una conexión del pool
58          conn = pool.getConnection();
59          stmt = conn.createStatement();
60          //Recuperar los parámetros usuario y password de la petición request
61          String usuario = request.getParameter("usuario");
62          String password = request.getParameter("password");
63          //Validar los parámetros de la petición request
64          if(usuario.length()==0)
65          {
66              //out.println("<h3>Debes introducir tu usuario</h3>");
67          }
68          else if(password.length()==0)
69          {
70              //out.println("<h3>Debes introducir tu contraseña</h3>");
71          }
72          else
73          {
74              //Verificar que existe el usuario y su correspondiente clave
75              StringBuilder sqlStr = new StringBuilder();
76              sqlStr.append("SELECT * FROM usuarios WHERE ");
77              sqlStr.append("STRCMP(usuarios.loginUsuario, ").append(usuario).append("'') = 0");
78              sqlStr.append(" AND STRCMP(usuarios.passUsuario, PASSWORD('").append(password).append("'')");
79              //out.println("<p>"+sqlStr.toString()+"</p>");
80              ResultSet rset = stmt.executeQuery(sqlStr.toString());
81              if(!rset.next())
82              {
83                  //Si el resultset no está vacío
84                  /*out.println("<h3>Nombre de usuario o contraseña incorrectos</h3>");
85                  out.println("<p><a href='index.html'>Volver a Login</a></p>");*/
86                  nextPage= "errorLogin.jsp";
87              }
88          }
89      }
90      else
91      {
92          //Si los datos introducidos son correctos

```

El código de nuestra página de entrada a la aplicación (index.html) es el siguiente:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link rel="stylesheet" type="text/css" href="./css/styleLogin.css">
<title>Login</title>
</head>
<body>
<h2>Acceso</h2>
<div class="login">
    <form method="post" action="login">
        <input type="text" name="usuario" placeholder="Username"
required="required" />
        <input type="password" name="password" placeholder="Password"
required="required" />
        <button type="submit" class="btn btn-primary btn-block btn-
large">Acceder</button>
        <button type="reset" class="btn btn-primary btn-block btn-
large">Cancelar</button>
    </form>
</div>
</body>
</html>
```

El resultado, una vez aplicado el CSS y el JS, sería:



El servlet que controla el acceso a nuestra aplicación web es LoginServlet.java:

```
package es.studium.LoginServlet;
import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.sql.DataSource;

import com.sun.istack.internal.logging.Logger;
/**
 * Servlet implementation class LoginServlet
 */
public class LoginServlet extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    // Pool de conexiones a la base de datos
    private DataSource pool;
    // Determina a qué página jsp se redirigirá
    String nextPage = "";
    /**
     * @see HttpServlet#HttpServlet()
     */
    public LoginServlet()
    {
        super();
    }
    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws
        ServletException, IOException
    {
        Connection conn = null;
        Statement stmt = null;
        try
```

```

{
    // Obtener una conexión del pool
    conn = pool.getConnection();
    stmt = conn.createStatement();
    //Recuperar los parámetros usuario y password de la petición
request
    String usuario = request.getParameter("usuario");
    String password = request.getParameter("password");
    //Validar los parámetros de la petición request
    if(usuario.length()==0)
    {

    }
    else if(password.length()==0)
    {

    }
    else
    {
        //Verificar que existe el usuario y su correspondiente
clave
        StringBuilder sqlStr = new StringBuilder();
        sqlStr.append("SELECT * FROM usuarios WHERE ");
        sqlStr.append("STRCMP(usuarios.loginUsuario,
'").append(usuario).append("' ) = 0");
        sqlStr.append(" AND STRCMP(usuarios.passUsuario,
PASSWORD('").append(password).append("' ) = 0");
        ResultSet rset = stmt.executeQuery(sqlStr.toString());
        if(!rset.next())
        {
            nextPage= "errorLogin.jsp";
        }
        else
        {
            HttpSession session = request.getSession(false);
            if(session != null)
            {
                session.invalidate();
            }
            session = request.getSession(true);
            synchronized(session)
            {
                session.setAttribute("usuario", usuario);
            }

            nextPage = "menu.jsp";
        }
    }
}

```



```

    }
    catch(SQLException ex)
    {

    }
    finally
    {
        try
        {
            if(stmt != null)
            {
                stmt.close();
            }
            if(conn != null)
            {
                // Esto devolvería la conexión al pool
                conn.close();
            }
        }
        catch(SQLException ex)
        {

        }
    }
    response.sendRedirect(nextPage);
}
/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException
{
    // TODO Auto-generated method stub
    doPost(request, response);
}
public void init(ServletConfig config) throws ServletException
{
    try
    {
        // Crea un contexto para poder luego buscar el recurso
DataSource
        InitialContext ctx = new InitialContext();
        // Busca el recurso DataSource en el contexto
        pool
        (DataSource)ctx.lookup("java:comp/env/jdbc/mysql_tiendaLibros");
    }
    catch (Exception e)
    {
        // TODO Auto-generated method stub
    }
}

```

```

        if(pool == null)
        {
            throw new ServletException("DataSource desconocida
'mysql_tiadalibros'");
        }
    }
    catch(NamingException ex)
    {
    }
}
}
}

```

En este fragmento del fichero “LoginServlet.java” (el servlet de login), podemos ver que iba a requerir una modificación casi completa. En la versión previa, este login se encargaba de generar una web. Fue mucho el código que tuvimos que comentar, antes de proceder a su eliminación. Esta tarea supuso tiempo de análisis, estudio, investigación y desarrollo. Además, como comentamos con anterioridad, el cifrado de las contraseñas, será uno de los puntos a tener en cuenta y que requerirán modificaciones en nuestro código.

Otro punto conflictivo y problemático en nuestro desarrollo fue el siguiente:



```

134     {
135         // Esto devolvería la conexión al pool
136         conn.close();
137     }
138 }
139 catch(SQLException ex)
140 {
141     Logger.getLogger(LoginServlet.class.getName(),
142         null).log(Level.SEVERE, null, ex);
143 }
144 }
145 //response.getWriter().append("Served at: ").append(request.getContextPath());
146
147 /*ServletContext servletContext = getServletContext();
148 RequestDispatcher requestDispatcher = servletContext.getRequestDispatcher(nextPage);
149 requestDispatcher.forward(request, response);*/
150
151 response.sendRedirect(nextPage);
152
153 }

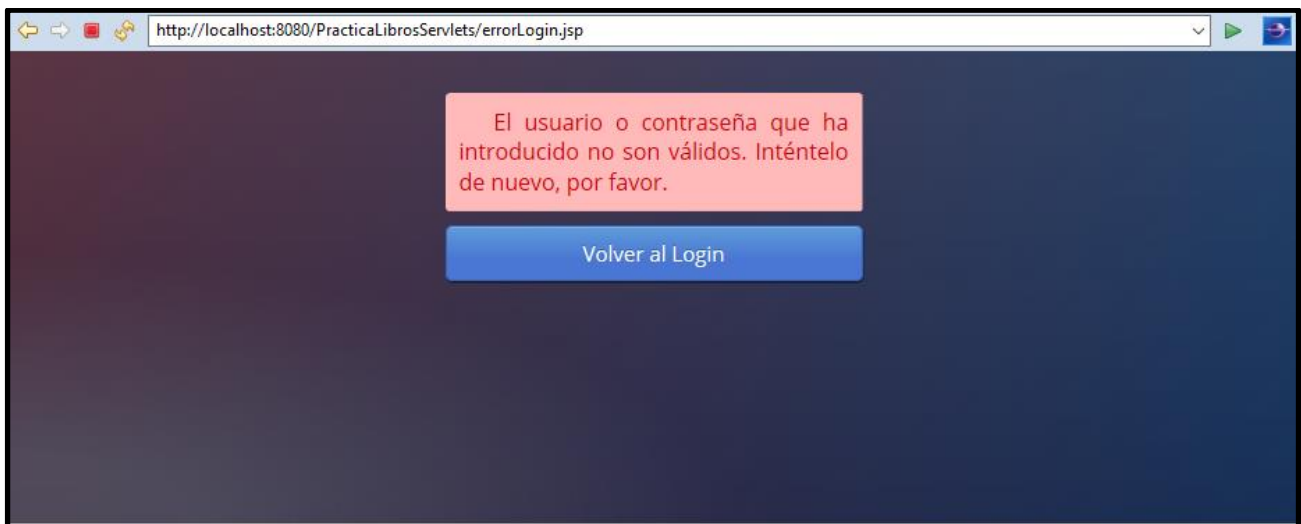
```

Tanto el Logger, como el ServletContext pusieron a prueba en mas de una ocasión nuestros conocimientos y nuestra paciencia con numerosos errores de toda índole. Por ello, buscamos soluciones alternativas para solventarlos.

En caso de que nuestro intento de login falle, seremos redireccionados a la página errorLogin.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Error de login</title>
<link rel="stylesheet" type="text/css" href="./css/styleLogin.css">
<link rel="stylesheet" type="text/css" href="./css/styleWarning.css">
</head>
<body>
    <div class="login">
        <div class="error-msg">
            <i class="fa fa-times-circle"></i> El usuario o contraseña que
ha introducido no son válidos. Inténtelo de nuevo, por favor.
        </div>
        <a href="./index.html" class="btn btn-primary btn-block btn-
large">Volver al Login</a>
    </div>
</body>
</html>
```

El resultado, aplicado el CSS y el JS, es el siguiente:



En el caso de que realicemos una validación correcta, seremos redirigidos a menu.jsp.

V. Aplicación Web

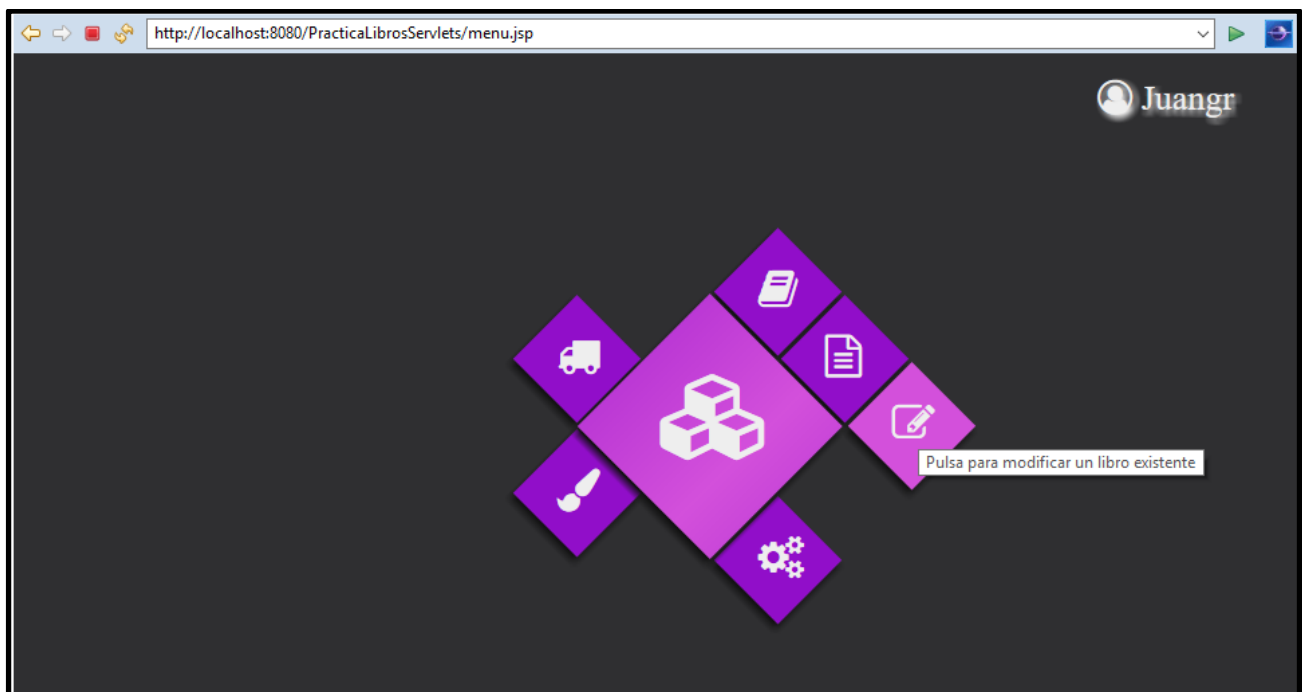
El archivo menu.jsp, nos permitirá la navegación entre los distintos elementos que componen nuestra aplicación. El código es el que se muestra a continuación:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Bienvenido al Menú</title>
<link rel='stylesheet' type='text/css' href='./css/styleMenu.css'>
<link rel="stylesheet" type="text/css"
    href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
<script src="./js/jquery-latest.js"></script>
<script type="text/javascript" src='./js/menu.js'></script>
</head>
<body>
    <div class="container">
        <div id="navMenu" class="diamond">
            <div id="mainRotCorrect" class="rotCorrect">
                <i class="fa fa-cubes fa-4x" aria-hidden="true"
                    title="Pulsa para abrir menú"></i>
            </div>
        </div>
        <div id="nav1" class="nav nav1 diamond">
            <div class="rotCorrect">
                <a href="/PracticaLibrosServlets/muestraLibro"><i
class="fa fa-book fa-2x" aria-hidden="true" title="Pulsa para consultar los
libros"></i></a>
            </div>
        </div>
        <div id="nav2" class="nav nav2 diamond">
            <div class="rotCorrect">
                <a href="vistas/insertLibro.jsp"><i class="fa fa-file-
text-o fa-2x" aria-hidden="true" title="Pulsa para insertar un nuevo libro"></i>
            </div>
        </div>
        <div id="nav3" class="nav nav3 diamond">
            <div class="rotCorrect">
                <a href="vistas/modificarLibro.jsp"><i class="fa fa-edit
fa-2x" aria-hidden="true" title="Pulsa para modificar un libro existente"></i>
            </div>
        </div>
        <div id="nav5" class="nav nav5 diamond">
```

```
<div class="rotCorrect">
    <a href="/PracticaLibrosServlets/muestraEditorial"><i
class="fa fa-cogs fa-2x" aria-hidden="true" title="Pulsa para consultar las
editoriales"></i></a>
</div>
</div>

<div id="nav7" class="nav nav7 diamond">
    <div class="rotCorrect">
        <a href="/PracticaLibrosServlets/muestraAutor"><i
class="fa fa-paint-brush fa-2x" aria-hidden="true" title="Pulsa para consultar
los autores"></i></a>
    </div>
</div>
<div id="nav8" class="nav nav8 diamond">
    <div class="rotCorrect">
        <a href="/PracticaLibrosServlets/muestraPedido"><i
class="fa fa-truck fa-2x" aria-hidden="true" title="Pulsa para consultar los
pedidos"></i></a>
    </div>
</div>
</div>
<div id="user"><i class="fa fa-user-circle-o fa-1x" aria-hidden="true"></i>
<%= (String)session.getAttribute("usuario") %></div>
</body>
</html>
```

El resultado que obtenemos al visualizar este .jsp, es el siguiente:



Nuestra aplicación web está construida según el Modelo-Vista-Controlador. Para cada una de las funcionalidades que implementa usamos un modelo, al menos una vista y un controlador.

A. Listar Autores

Comenzaremos a mostrar los autores. Hemos empleado el modelo Autor.java, la vista listAutor.jsp y el controlador (servlet) muestraAutor.java.

El modelo está formado por el código:

```
package es.studium.modelo;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import javax.naming.InitialContext;
import javax.sql.DataSource;

public class Autor {
    private int idAutor;
    private String nombreAutor;
    private String apellido1Autor;
    private String apellido2Autor;
    private DataSource pool;

    public Autor()
    {
        this.idAutor = 0;
        this.nombreAutor = null;
        this.apellido1Autor = null;
        this.apellido2Autor = null;
    }

    public Autor(int idAutor, String nombreAutor, String apellido1Autor, String
apellido2Autor){
        this.idAutor = idAutor;
        this.nombreAutor = nombreAutor;
        this.apellido1Autor = apellido1Autor;
        this.apellido2Autor = apellido2Autor;
    }

    public int getIdAutor() {
        return idAutor;
    }

    public void setIdAutor(int idAutor) {
```

```
        this.idAutor = idAutor;
    }

    public String getNombreAutor() {
        return nombreAutor;
    }

    public void setNombreAutor(String nombreAutor) {
        this.nombreAutor = nombreAutor;
    }

    public String getApellido1Autor() {
        return apellido1Autor;
    }

    public void setApellido1Autor(String apellido1Autor) {
        this.apellido1Autor = apellido1Autor;
    }

    public String getApellido2Autor() {
        return apellido2Autor;
    }

    public void setApellido2Autor(String apellido2Autor) {
        this.apellido2Autor = apellido2Autor;
    }

    public List<Autor> listarAutor()
    {
        List<Autor> listaAutores = new ArrayList<Autor>();
        String sql = "SELECT * FROM autores";
        try
        {
            Connection conn = null;
            Statement stmt = null;
            InitialContext ctx = new InitialContext();
            pool = (DataSource)ctx.lookup("java:comp/env/jdbc/mysql_tiendalibros");
            conn = pool.getConnection();

            stmt = conn.createStatement();
            ResultSet resultSet = stmt.executeQuery(sql);
            while (resultSet.next())
            {
                int idAutor = resultSet.getInt("idAutor");
                String nombreAutor = resultSet.getString("nombreAutor");
                String apellido1Autor = resultSet.getString("apellido1Autor");
                String apellido2Autor = resultSet.getString("apellido2Autor");
                Autor autor = new Autor(idAutor, nombreAutor, apellido1Autor, apellido2Autor);
                listaAutores.add(autor);
            }
        }
        catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
}
```

```

        String apellido2Autor =
resultSet.getString("apellido2Autor");

        Autor autor = new Autor(idAutor, nombreAutor,
apellido1Autor, apellido2Autor);
        listaAutores.add(autor);
    }

    }
    catch (Exception E)
    {
        System.out.println(E.getMessage()+E.getStackTrace());
    }
    return listaAutores;
}
}

```

El servlet que actúa como controlador (MuestraAutor.java) es el siguiente:

```

package es.studium.LoginServlet;
import java.io.IOException;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import es.studium.modelo.Autor;
/**
 * Servlet implementation class HazAlgo
 */
public class MuestraAutor extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public MuestraAutor()
    {
        super();
        // TODO Auto-generated constructor stub
    }
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws

```



```

ServletException, IOException
{
    try
    {
        HttpSession session = request.getSession(false);
        synchronized(session)
        {
            Autor autor = new Autor();
            List <Autor> autores = autor.listarAutor();
            session.setAttribute("autores", autores);
            response.sendRedirect("../vistas/listAutor.jsp");
        }

    }catch(Exception E){
        System.out.println(E.getMessage());
    }

    response.getWriter().append("Served at:
").append(request.getContextPath());
}
/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException
{
    // TODO Auto-generated method stub
    doPost(request, response);
}
}

```

La vista que emplearemos para mostrar los autores (listAutor.jsp) es la siguiente:

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page session="true" import="java.util.*, es.studium.modelo.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//ES"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Listado de Autores</title>
<link rel="stylesheet" type="text/css" href="../css/styleInsert.css">
<link
        rel="stylesheet"
        type="text/css"
href="https://cdn.datatables.net/1.10.16/css/jquery.dataTables.min.css">
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>

```

```

<script
src="https://cdn.datatables.net/1.10.16/js/jquery.dataTables.min.js"></script>
<script type="text/javascript">
$(document).ready(function() {
    $('#example').DataTable();
} );
</script>
</head>
<body>
    <h1>Autores MVC: Listado de autores</h1>
    <hr />
    <h2>Este es el listado de los autores en la base de datos:</h2>
    <table id="example" class="display" cellspacing="0">
        <thead>
            <tr>
                <th>idAutor</th>
                <th>NombreAutor</th>
                <th>Apellido1Autor</th>
                <th>Apellido2Autor</th>
            </tr>
        </thead>
        <tbody>
            <%
                List<Autor> listaAutores = (List<Autor>)
session.getAttribute("autores");
                for (Autor autor : listaAutores) {
                    %>
                    <tr>
                        <td align="right"><%=autor.getIdAutor()%></td>
                        <td><%=autor.getNombreAutor()%></td>
                        <td><%=autor.getApellido1Autor()%></td>
                        <td><%=autor.getApellido2Autor()%></td>
                    </tr>

                    <%
                        }
                    %>
                </tbody>
            </table>
            <br />
            <a class="button" href="../menu.jsp">Volver al menú</a>
            <br/><br/>
            <a class="button" href="../logout">Cerrar la sesión</a>
        </body>
    </html>

```

El resultado obtenido es:

Autores MVC: Listado de autores

Este es el listado de los autores en la base de datos:

Show entries Search:

| idAutor | NombreAutor | Apellido1Autor | Apellido2Autor |
|---------|-------------|----------------|----------------|
| 1 | George | R.R. | Martin |
| 2 | J | R.R. | Tolkien |
| 3 | Arturo | Pérez | Reverte |
| 4 | Gabriel | García | Márquez |
| 5 | Miguel | de Cervantes | Saavedra |
| 6 | Mario | Vargas | Llosa |
| 7 | Mario | Benedetti | Farugia |
| 8 | Federico | García | Lorca |
| 9 | Isabel | Allende | Llona |
| 10 | Agatha | Christie | - |

Showing 1 to 10 of 13 entries Previous 2 Next

[Volver al menú](#)

[Cerrar la sesión](#)

Adicionalmente, comentar que, si hacemos clic sobre el enlace “Cerrar la sesión”, seremos redirigidos al servlet LogoutServlet.java. Este servlet, cerrará las invalidará la sesión y nos remitirá a index.html para que podamos volver a acceder a la aplicación:

```
package es.studium.LoginServlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
/**
 * Servlet implementation class LogoutServlet
 */
public class LogoutServlet extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
    String nextPage = "";

    public LogoutServlet()
    {
        super();
        // TODO Auto-generated constructor stub
    }
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
    response)
```

```
        */
        protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws
        ServletException, IOException
        {
            try
            {
                HttpSession session = request.getSession(false);
                nextPage= "index.html";
                if(session == null)
                {

                }
                else
                {
                    session.invalidate();
                    response.sendRedirect(nextPage);
                }
            }
            finally
            {

            }
        }
    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws
        ServletException, IOException
    {
        // TODO Auto-generated method stub
        doPost(request, response);
    }
}
```

B. Listar Editoriales

La estructura que emplearemos, es similar a la utilizada en Autores. Usaremos el Modelo-Vista-Controlador. Pasamos a detallar pormenorizadamente, cada uno de los archivos, comenzando por Editorial.java:

```
package es.studium.modelo;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import javax.naming.InitialContext;
import javax.sql.DataSource;

public class Editorial {

    private int idEditorial;
    private String nombreEditorial;
    private DataSource pool;

    public Editorial(){
        this.idEditorial = 0;
        this.nombreEditorial = null;
    }

    public Editorial(int idEditorial, String nombreEditorial){
        this.idEditorial = idEditorial;
        this.nombreEditorial = nombreEditorial;
    }

    public int getIdEditorial() {
        return idEditorial;
    }

    public void setIdEditorial(int idEditorial) {
        this.idEditorial = idEditorial;
    }

    public String getNombreEditorial() {
        return nombreEditorial;
    }

    public void setNombreEditorial(String nombreEditorial) {
        this.nombreEditorial = nombreEditorial;
    }

    public List<Editorial> listarEditorial()
```

```

{
    List<Editorial> listaEditoriales = new ArrayList<Editorial>();
    String sql = "SELECT * FROM editoriales";
    // Aquí deberíamos de hacer la conexión
    try
    {
        Connection conn = null;
        Statement stmt = null;

        InitialContext ctx = new InitialContext();
        pool = (DataSource)ctx.lookup("java:comp/env/jdbc/mysql_tiendalibros");
        conn = pool.getConnection();
        stmt = conn.createStatement();
        ResultSet resultSet = stmt.executeQuery(sql);
        //Recorremos la base de datos y mientras queden resultados
        realizaremos lo siguiente
        while (resultSet.next())
        {
            int idEditorial = resultSet.getInt("idEditorial");
            String nombreEditorial = resultSet.getString("nombreEditorial");
            Editorial editorial = new Editorial(idEditorial, nombreEditorial);
            listaEditoriales.add(editorial);
        }
    }
    catch (Exception E)
    {
        System.out.println(E.getMessage()+E.getStackTrace());
    }
    return listaEditoriales;
}
}

```

El servlet que controlará el listado de editoriales es MuestraEditorial.java y su código es el siguiente:

```

package es.studium.LoginServlet;
import java.io.IOException;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

```

```

import es.studium.modelo.Editorial;
/**
 * Servlet implementation class HazAlgo
 */
public class MuestraEditorial extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public MuestraEditorial()
    {
        super();
        // TODO Auto-generated constructor stub
    }
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException
    {
        try
        {
            HttpSession session = request.getSession(false);
            synchronized(session)
            {
                Editorial editorial = new Editorial();
                List<Editorial> editoriales =
editorial.listarEditorial();
                session.setAttribute("editoriales", editoriales);

                response.sendRedirect("./vistas/listEditorial.jsp");

            }
        }catch(Exception E){
            System.out.println(E.getMessage());
        }
        response.getWriter().append("Served at:
").append(request.getContextPath());
    }
    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws

```

```

        ServletException, IOException
    {
        doPost(request, response);
    }
}

```

Por último, la vista encargada de mostrar las editoriales es listEditorial.jsp:

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page session="true" import="java.util.*, es.studium.modelo.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//ES"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Listado de Editoriales</title>
<link rel="stylesheet" type="text/css" href="../css/styleInsert.css">
<link rel="stylesheet" type="text/css"
    href="https://cdn.datatables.net/1.10.16/css/jquery.dataTables.min.css">
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
<script
    src="https://cdn.datatables.net/1.10.16/js/jquery.dataTables.min.js"></sc
ript>
<script type="text/javascript">
    $(document).ready(function() {
        $('#example').DataTable();
    });
</script>
</head>
<body>
    <h1>Editoriales MVC: Listado de editoriales</h1>
    <hr />
    <h2>Este es el listado de las editoriales en la base de datos:</h2>
    <table id="example" class="display" cellspacing="0">
        <thead>
            <tr>
                <th>idEditorial</th>
                <th>NombreEditorial</th>
            </tr>
        </thead>
        <tbody>

            <%
                List<Editorial> listaEditoriales = (List<Editorial>)
session.getAttribute("editoriales");
                for (Editorial editorial : listaEditoriales) {
                    %>

```



```

        <tr>
            <td align="right"><%=editorial.getIdEditorial()%></td>
            <td><%=editorial.getNombreEditorial()%></td>
        </tr>

    <%
        }
        //session.invalidate();
    %>
</tbody>
</table>
<br />
<a class="button" href="../menu.jsp">Volver al menú</a>
<br/><br/>
<a class="button" href="../logout">Cerrar la sesión</a>
</body>
</html>

```

El resultado obtenido al visualizar el jsp es el siguiente:

| Editoriales MVC: Listado de editoriales | | |
|--|--|--|
| Este es el listado de las editoriales en la base de datos: | | |
| Show <input type="text" value="10"/> entries | | Search: <input type="text"/> |
| idEditorial | | NombreEditorial |
| 1 | | Gilgamesh |
| 2 | | Alianza Editorial |
| 3 | | Editorial Planeta |
| Showing 1 to 3 of 3 entries | | Previous <input type="text" value="1"/> Next |
| <input type="button" value="Volver al menú"/> | | |
| <input type="button" value="Cerrar la sesión"/> | | |

C. Listar pedidos y modificar fecha de envío

Al igual que en los dos casos anteriores, el código empleado para generar el listado es muy similar. En este, nos encontramos con la diferencia de que debemos ser capaces de modificar los valores de la base de datos. Para ello, vamos a utilizar Modelo-Vista-Controlador.

El modelo empleado es Pedido.java:

```

package es.studium.modelo;

import java.sql.Connection;
import java.sql.Date;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

```

```
import javax.naming.InitialContext;
import javax.sql.DataSource;

public class Pedido {

    private int idPedido;
    private Date fechaEnvioPedido;
    private Date fechaPedido;
    private int cantidadPedido;
    private String loginUsuario;
    private String tituloLibro;
    private DataSource pool;
    private int idUsuarioFK;
    private int idLibroFK;

    public Pedido(){
        this.idPedido = 0;
        this.fechaEnvioPedido = null;
        this.fechaPedido = null;
        this.cantidadPedido = 0;
        this.loginUsuario = null;
        this.tituloLibro = null;
        this.idUsuarioFK = 0;
        this.idLibroFK = 0;
    }

    public Pedido (int idPedido){
        this.idPedido = idPedido;
    }

    public Pedido (int idPedido, Date fechaEnvioPedido, Date fechaPedido, int
cantidadPedido, String loginUsuario, String tituloLibro){
        this.idPedido = idPedido;
        this.fechaEnvioPedido = fechaEnvioPedido;
        this.fechaPedido = fechaPedido;
        this.cantidadPedido = cantidadPedido;
        this.loginUsuario = loginUsuario;
        this.tituloLibro = tituloLibro;
    }

    public int getIdPedido() {
        return idPedido;
    }

    public void setIdPedido(int idPedido) {
        this.idPedido = idPedido;
    }
}
```

```
public Date getFechaEnvioPedido() {  
    return fechaEnvioPedido;  
}  
  
public void setFechaEnvioPedido(Date fechaEnvioPedido) {  
    this.fechaEnvioPedido = fechaEnvioPedido;  
}  
  
public Date getFechaPedido() {  
    return fechaPedido;  
}  
  
public void setFechaPedido(Date fechaPedido) {  
    this.fechaPedido = fechaPedido;  
}  
  
public int getCantidadPedido() {  
    return cantidadPedido;  
}  
  
public void setCantidadPedido(int cantidadPedido) {  
    this.cantidadPedido = cantidadPedido;  
}  
  
public String getLoginUsuario() {  
    return loginUsuario;  
}  
  
public void setLoginUsuario(String loginUsuario) {  
    this.loginUsuario = loginUsuario;  
}  
  
public String getTituloLibro() {  
    return tituloLibro;  
}  
  
public void setTituloLibro(String tituloLibro) {  
    this.tituloLibro = tituloLibro;  
}  
  
public int getIdUsuarioFK() {  
    return idUsuarioFK;  
}  
  
public void setIdUsuarioFK(int idUsuarioFK) {  
    this.idUsuarioFK = idUsuarioFK;  
}
```

```

public int getIdLibroFK() {
    return idLibroFK;
}

public void setIdLibroFK(int idLibroFK) {
    this.idLibroFK = idLibroFK;
}

public List<Pedido> listarPedido()
{
    List<Pedido> listaPedidos = new ArrayList<Pedido>();
    String sql = "SELECT
idPedido,fechaEnvioPedido,fechaPedido,cantidadPedido,loginUsuario,tituloLibro "
        + "FROM Pedidos INNER JOIN Usuarios ON Pedidos.idUsuarioFK
= usuarios.idUsuario "
        + "INNER JOIN Libros ON Pedidos.idLibroFK =
Libros.idLibro;";
    // Aquí deberíamos de hacer la conexión
    try
    {
        Connection conn = null;
        Statement stmt = null;

        InitialContext ctx = new InitialContext();
        pool =
        (DataSource)ctx.lookup("java:comp/env/jdbc/mysql_tiadalibros");
        conn = pool.getConnection();
        //Creamos el statement
        stmt = conn.createStatement();
        ResultSet resultSet = stmt.executeQuery(sql);
        //Recorremos la base de datos y mientras queden resultados
        realizaremos lo siguiente
        while (resultSet.next())
        {
            int idPedido = resultSet.getInt("idPedido");
            Date fechaEnvioPedido =
resultSet.getDate("fechaEnvioPedido");
            Date fechaPedido = resultSet.getDate("fechaPedido");
            int cantidadPedido = resultSet.getInt("cantidadPedido");
            String loginUsuario = resultSet.getString("loginUsuario");
            String tituloLibro = resultSet.getString("tituloLibro");
            Pedido pedido = new Pedido(idPedido, fechaEnvioPedido,
fechaPedido, cantidadPedido, loginUsuario, tituloLibro);
            listaPedidos.add(pedido);
        }
    }
}

```

```
        catch (Exception E)
        {
            System.out.println(E.getMessage()+E.getStackTrace());
        }
        return listaPedidos;
    }

    public boolean modificarPedido(Pedido pedido) throws SQLException {

        try
        {
            Connection conn = null;
            Statement stmt = null;
            InitialContext ctx = new InitialContext();
            pool = (DataSource)ctx.lookup("java:comp/env/jdbc/mysql_tiendalibros");
            conn = pool.getConnection();

            stmt = conn.createStatement();

            String sql = "UPDATE pedidos SET fechaEnvioPedido= CURDATE()
"+"WHERE idPedido="+pedido.getIdPedido()+"";

            int resultado = stmt.executeUpdate(sql);
            stmt.close();
            return true;
        }
        catch (Exception E)
        {
            System.out.println(E.getMessage()+E.getStackTrace());
        }
        return false;
    }
}
```

Una vez, realizado el código del modelo, pasamos a los servlets. En este caso serán dos los encargados de gestionar la vista. El primero de ellos, MuestraPedido.java se encarga de listar los pedidos. El segundo, ModificarPedido.java, es el encargado de la gestión del cambio de fecha de envío.

MuestraPedido.java posee el siguiente código:

```
package es.studium.LoginServlet;
import java.io.IOException;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import es.studium.modelo.Pedido;
/**
 * Servlet implementation class HazAlgo
 */
public class MuestraPedido extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public MuestraPedido()
    {
        super();
        // TODO Auto-generated constructor stub
    }
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException
    {

        try
        {

            HttpSession session = request.getSession(false);

            synchronized(session)
            {

                Pedido pedido = new Pedido();

                List <Pedido> pedidos = pedido.listarPedido();
```

```
        session.setAttribute("pedidos", pedidos);

        response.sendRedirect("./vistas/listPedido.jsp");

    }

    }catch(Exception E){
        System.out.println(E.getMessage());
    }

}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException
{

    doPost(request, response);
}
}
```

Por otro lado, ModificarPedido, está compuesto por el siguiente código:

```
package es.studium.LoginServlet;
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import es.studium.modelo.Pedido;
/**
 * Servlet implementation class HazAlgo
 */
public class ModificarPedido extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public ModificarPedido()
    {
        super();
    }
}
```

```
// TODO Auto-generated constructor stub
}
/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException
{
    try
    {
        HttpSession session = request.getSession(false);
        synchronized(session)
        {
            String todo = request.getParameter("todo");
            if(todo==null)
            {

            }
            else if(todo.equals("modify"))
            {
                Pedido nuevoPedido = new Pedido(

Integer.parseInt(request.getParameter("idPedido")));

                if(nuevoPedido.modificarPedido(nuevoPedido))
                {

response.sendRedirect("/PracticaLibrosServlets/muestraPedido");
                }
                else
                {

                }

            }
        }
    }
    catch(Exception E){
        System.out.println(E.getMessage());
    }
}
/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
```



```
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException
{
    // TODO Auto-generated method stub
    doPost(request, response);
}
}
```

Como hemos visto a lo largo de todos los ejemplos mostrados, el código de nuestros servlets es bastante sencillo, ya que la lógica de los mismos está remitida a funciones en los modelos. Esto simplifica mucho la comprensión de los mismos.

Por último, pero no por ello menos importante, el código de listPedido.jsp, la vista encargada de mostrarnos la “magia”:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page session="true" import="java.util.*, es.studium.modelo.*"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Listado de Pedidos</title>
<link rel="stylesheet" type="text/css" href="../css/styleInsert.css">
<link rel="stylesheet" type="text/css"
    href="https://cdn.datatables.net/1.10.16/css/jquery.dataTables.min.css">
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
<script
    src="https://cdn.datatables.net/1.10.16/js/jquery.dataTables.min.js"></sc
ript>
<script type="text/javascript">
    $(document).ready(function() {
        $('#example').DataTable();
    });
</script>
</head>
<body>
    <h1>Pedidos MVC: Listado de pedidos</h1>
    <hr />
    <h2>Este es el listado de los pedidos en la base de datos:</h2>
    <table id="example" class="display">
        <thead>
            <tr>
                <th>idPedido</th>
                <th>fechaEnvioPedido</th>
                <th>fechaPedido</th>
                <th>cantidadPedido</th>
```

```

        <th>loginUsuario</th>
        <th>tituloLibro</th>
    </tr>
</thead>
<%
    //TODO Muestra los elementos del carrito
    List<Pedido> listaPedidos = (List<Pedido>)
session.getAttribute("pedidos");
    for (Pedido pedido : listaPedidos) {
        %>

        <tbody>
            <tr>
                <td align="right"><%=pedido.getIdPedido()%></td>
                <td><%=pedido.getFechaEnvioPedido()%></td>
                <td><%=pedido.getFechaPedido()%></td>
                <td><%=pedido.getCantidadPedido()%></td>
                <td><%=pedido.getLoginUsuario()%></td>
                <td><%=pedido.getTituloLibro()%></td>
            </tr>
        </tbody>
    <%
    }

    %>

</table>
<br />
<hr />
<h2>Inserte idLibro para marcar como enviado</h2>
<form name="modificarPed" action="../modificarPedido" method="POST">
    <input type="hidden" id="todo" name="todo" value="modify">
    <div class="form1">
        <label for="idPedido" id="idPedido1">idPedido<br /></label>
    </div>
    <div class="form2">
        <select name="idPedido">
            <%
                for (Pedido pedido : listaPedidos) {
                    // Scriplet 1: Carg los libros en el SELECT

                    out.println("<option          value='"          +
pedido.getIdPedido() + "'>");
                    out.println(pedido.getIdPedido()+ " | " +
pedido.getLoginUsuario()
                    + " | " + pedido.getTituloLibro());
                    out.println("</option>");
                }
            %>
        </select>
    </div>
</form>

```

```

        %>
    </select>
</div>
<br /> <input id="submitButton" type="submit"
        value="Modificar pedido">
</form>
<br />
<a class="button" href=" ../menu.jsp">Volver al menú</a>
<br/><br/>
<a class="button" href=" ../logout">Cerrar la sesión</a>
</body>
</html>

```

El resultado que obtenemos al visualizar el jsp es:

Pedidos MVC: Listado de pedidos

Este es el listado de los pedidos en la base de datos:

Show 10 entries

Search:

| idPedido | fechaEnvioPedido | fechaPedido | cantidadPedido | loginUsuario | tituloLibro |
|----------|------------------|-------------|----------------|--------------|-------------------------|
| 1 | 2018-03-06 | 2018-01-26 | 1 | juangr | Juego de Tronos |
| 2 | 2018-03-08 | 2018-01-28 | 1 | mariapg | El Capitán Alatriste |
| 3 | 2018-03-06 | 2018-01-29 | 1 | juangr | El Señor de los Anillos |

Showing 1 to 1 of 1 entries

Previous
1
Next

Inserte idLibro para marcar como enviado

idPedido

1 | juangr | Juego de Tronos

Modificar pedido

Volver al menú

Cerrar la sesión

D. Listado de libros

La clase Libro.java es el modelo común que emplearán todos los Modelo-Vista-Controlador que interactúan con los libros. Por ello, sólo vamos a desglosarlo una vez, para no alargar innecesariamente esta práctica. El modelo contiene en su interior todas las funciones que llamaremos desde el resto de servlets y facilitará notablemente su comprensión.

El código con el que generamos el modelo Libro.java es:

```
package es.studium.modelo;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

import javax.naming.InitialContext;
import javax.sql.DataSource;

public class Libro {

    private int idLibro;
    private String tituloLibro;
    private double precioLibro;
    private int cantidadLibro;
    private int idEditorialFK;
    private int idAutorFK;
    private String nombreEditorial;
    private String nombreAutor;
    private String apellido1Autor;
    private String apellido2Autor;
    private DataSource pool;

    public Libro(){
        this.idLibro = 0;
        this.tituloLibro= null;
        this.precioLibro = 0.0;
        this.cantidadLibro= 0;
        this.idEditorialFK = 0;
        this.idAutorFK= 0;
    }

    public Libro(int idLibro, String tituloLibro, double precioLibro, int
cantidadLibro, int idEditorialFK, int idAutorFK){
        this.idLibro = idLibro;
        this.tituloLibro= tituloLibro;
        this.precioLibro = precioLibro;
        this.cantidadLibro= cantidadLibro;
        this.idEditorialFK = idEditorialFK;
        this.idAutorFK= idAutorFK;
    }
}
```

```
public Libro(int idLibro, String tituloLibro, double precioLibro, int
cantidadLibro, String nombreEditorial, String nombreAutor, String apellido1Autor,
String apellido2Autor){
    this.idLibro = idLibro;
    this.tituloLibro= tituloLibro;
    this.precioLibro = precioLibro;
    this.cantidadLibro= cantidadLibro;
    this.nombreEditorial = nombreEditorial;
    this.nombreAutor = nombreAutor;
    this.apellido1Autor = apellido1Autor;
    this.apellido2Autor = apellido2Autor;

}

public int getIdLibro() {
    return idLibro;
}

public void setIdLibro(int idLibro) {
    this.idLibro = idLibro;
}

public String getTituloLibro() {
    return tituloLibro;
}

public void setTituloLibro(String tituloLibro) {
    this.tituloLibro = tituloLibro;
}

public double getPrecioLibro() {
    return precioLibro;
}

public void setPrecioLibro(double precioLibro) {
    this.precioLibro = precioLibro;
}

public int getCantidadLibro() {
    return cantidadLibro;
}

public void setCantidadLibro(int cantidadLibro) {
    this.cantidadLibro = cantidadLibro;
}

public int getIdEditorialFK() {
    return idEditorialFK;
}
```

```
}

public void setIdEditorialFK(int idEditorialFK) {
    this.idEditorialFK = idEditorialFK;
}

public int getIdAutorFK() {
    return idAutorFK;
}

public void setIdAutorFK(int idAutorFK) {
    this.idAutorFK = idAutorFK;
}

public String getNombreEditorial() {
    return nombreEditorial;
}

public void setNombreEditorial(String nombreEditorial) {
    this.nombreEditorial = nombreEditorial;
}

public String getNombreAutor() {
    return nombreAutor;
}

public void setNombreAutor(String nombreAutor) {
    this.nombreAutor = nombreAutor;
}

public String getApellido1Autor() {
    return apellido1Autor;
}

public void setApellido1Autor(String apellido1Autor) {
    this.apellido1Autor = apellido1Autor;
}

public String getApellido2Autor() {
    return apellido2Autor;
}

public void setApellido2Autor(String apellido2Autor) {
    this.apellido2Autor = apellido2Autor;
}

public List<Libro> listarLibro()
{
```

```

        List<Libro> listaLibros = new ArrayList<Libro>();

        String sql = "SELECT
idLibro,tituloLibro,precioLibro,cantidadLibro,nombreEditorial,nombreAutor,apell
ido1Autor,apellido2Autor "
            + "from Libros INNER JOIN Autores ON Libros.idAutorFK =
Autores.idAutor INNER JOIN Editoriales ON Libros.idEditorialFK "
            + "= Editoriales.idEditorial";

        try
        {
            Connection conn = null;
            Statement stmt = null;

            InitialContext ctx = new InitialContext();
            pool =
(DataSource)ctx.lookup("java:comp/env/jdbc/mysql_tiadalibros");
            conn = pool.getConnection();

            stmt = conn.createStatement();

            ResultSet resultSet = stmt.executeQuery(sql);

            while (resultSet.next())
            {
                int idLibro = resultSet.getInt("idLibro");
                String tituloLibro = resultSet.getString("tituloLibro");
                Double precioLibro = resultSet.getDouble("precioLibro");
                int cantidadLibro = resultSet.getInt("cantidadLibro");
                String nombreEditorial =
resultSet.getString("nombreEditorial");
                String nombreAutor = resultSet.getString("nombreAutor");
                String apellido1Autor =
resultSet.getString("apellido1Autor");
                String apellido2Autor =
resultSet.getString("apellido2Autor");
                Libro libro = new Libro(idLibro, tituloLibro,
precioLibro, cantidadLibro, nombreEditorial, nombreAutor, apellido1Autor,
apellido2Autor);
                listaLibros.add(libro);
            }
        }
        catch (Exception E)
        {
            System.out.println(E.getMessage()+E.getStackTrace());
        }
    }
}

```

```

        return listaLibros;
    }

    public boolean insertarLibro(Libro libro) throws SQLException {

        try
        {
            Connection conn = null;
            Statement stmt = null;
            InitialContext ctx = new InitialContext();
            pool = (DataSource)ctx.lookup("java:comp/env/jdbc/mysql_tiendalibros");
            conn = pool.getConnection();
            stmt = conn.createStatement();
            String sql2 = "INSERT INTO libros VALUES
(null, '"+libro.getTituloLibro()+"', '"+
libro.getPrecioLibro()+"', '"+libro.getCantidadLibro()+"', '"+libro.getIdEditorialFK()+"', '"+libro.getIdAutorFK()+"');";
            int resultado = stmt.executeUpdate(sql2);
            stmt.close();
            return true;
        }
        catch (Exception E)
        {
            System.out.println(E.getMessage()+E.getStackTrace());
        }
        return false;
    }

    public boolean modificarLibro(Libro libro) throws SQLException {

        try
        {
            Connection conn = null;
            Statement stmt = null;
            InitialContext ctx = new InitialContext();
            pool = (DataSource)ctx.lookup("java:comp/env/jdbc/mysql_tiendalibros");
            conn = pool.getConnection();
            stmt = conn.createStatement();
            String sql3 = "UPDATE libros SET
tituloLibro='"+libro.getTituloLibro()+"',
precioLibro='"+libro.getPrecioLibro()+"',
cantidadLibro='"+libro.getCantidadLibro()+"',

```



```

idEditorialFK="+libro.getIdEditorialFK()+", idAutorFK="+libro.getIdAutorFK()+"
WHERE idlibro="+libro.getIdLibro()+"";
        int resultado = stmt.executeUpdate(sql3);
        stmt.close();
        return true;
    }
    catch (Exception E)
    {
        System.out.println(E.getMessage()+E.getStackTrace());
    }
    return false;
}
}

```

El servlet que gestionará el listado de los libros en la base de datos (MuestraLibro.java) está formado por el siguiente código:

```

package es.studium.LoginServlet;
import java.io.IOException;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import es.studium.modelo.Libro;
/**
 * Servlet implementation class HazAlgo
 */
public class MuestraLibro extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public MuestraLibro()
    {
        super();
        // TODO Auto-generated constructor stub
    }
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
     */

```

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException
{
    try
    {
        HttpSession session = request.getSession(false);
        synchronized(session)
        {

            Libro libro = new Libro();

            List <Libro> libros = libro.listarLibro();

            session.setAttribute("libros", libros);

            response.sendRedirect("./vistas/listLibro.jsp");

        }

    }catch(Exception E){
        System.out.println(E.getMessage());
    }

    response.getWriter().append("Served at:
").append(request.getContextPath());
}
/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException
{
    // TODO Auto-generated method stub
    doPost(request, response);
}
}
```

La vista listLibro.jsp tiene el siguiente código:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page session="true" import="java.util.*, es.studium.modelo.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//ES"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Listado de Libros</title>
<link rel="stylesheet" type="text/css" href="../css/styleInsert.css">
<link rel="stylesheet" type="text/css"
    href="https://cdn.datatables.net/1.10.16/css/jquery.dataTables.min.css">
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
<script
    src="https://cdn.datatables.net/1.10.16/js/jquery.dataTables.min.js"></sc
ript>
<script type="text/javascript">
    $(document).ready(function() {
        $('#example').DataTable();
    });
</script>
</head>
<body>
    <h1>Libros MVC: Listado de libros</h1>
    <hr />
    <h2>Este es el listado de los libros en la base de datos:</h2>
    <table id="example" class="display" cellspacing="0">
        <thead>
            <tr>
                <th>idLibro</th>
                <th>tituloLibro</th>
                <th>precioLibro</th>
                <th>cantidadLibro</th>
                <th>nombreEditorial</th>
                <th>NombreAutor</th>
                <th>Apellido1Autor</th>
                <th>Apellido2Autor</th>
            </tr>
        </thead>
        <tbody>

            <%
                //TODO Muestra los elementos del carrito
                List<Libro> listaLibros = (List<Libro>)
session.getAttribute("libros");
                for (Libro libro : listaLibros) {
```

```

        %>
        <tr>
            <td align="right"><%=libro.getIdLibro()%></td>
            <td><%=libro.getTituloLibro()%></td>
            <td align="right"><%=libro.getPrecioLibro()%></td>
            <td align="right"><%=libro.getCantidadLibro()%></td>
            <td><%=libro.getNombreEditorial()%></td>
            <td><%=libro.getNombreAutor()%></td>
            <td><%=libro.getApellido1Autor()%></td>
            <td><%=libro.getApellido2Autor()%></td>
        </tr>

    <%
    }

    %>

</tbody>
</table>
<br />
<a class="button" href="../menu.jsp">Volver al menú</a>
<br/><br/>
<a class="button" href="../logout">Cerrar la sesión</a>

</body>
</html>

```

El resultado obtenido a partir de la vista es:

| Libros MVC: Listado de libros | | | | | | | | |
|---|--------------------------|-------------|---------------|-------------------|-------------|----------------|----------------|--------|
| Este es el listado de los libros en la base de datos: | | | | | | | | |
| Show | 10 | ▼ | entries | | | | | |
| | | | | | | Search: | | |
| idLibro | tituloLibro | precioLibro | cantidadLibro | nombreEditorial | NombreAutor | Apellido1Autor | Apellido2Autor | |
| 1 | Juego de Tronos | 25.99 | 10 | Gilgamesh | George | R.R. | Martin | |
| 2 | El Señor de los Anillos | 22.99 | 10 | Alianza Editorial | J | R.R. | Tolkien | |
| 3 | El Capitán Alatriste | 23.99 | 10 | Editorial Planeta | Arturo | Pérez | Reverte | |
| 4 | Ignacio el valiente | 11.99 | 7 | Gilgamesh | George | R.R. | Martin | |
| 5 | El dragón de fuego | 12.95 | 5 | Gilgamesh | George | R.R. | Martin | |
| 6 | fray perico y su borrico | 12.98 | 9 | Alianza Editorial | J | R.R. | Tolkien | |
| 7 | Fray Quisquilla | 13.44 | 1 | Alianza Editorial | George | R.R. | Martin | |
| 8 | El dragon de hielo | 11.11 | 2 | Gilgamesh | Arturo | Pérez | Reverte | |
| 9 | Pepito de los palotes | 10.0 | 5 | Editorial Planeta | Agatha | Christie | - | |
| 10 | Godofredo el Feo | 11.11 | 6 | Gilgamesh | George | R.R. | Martin | |
| Showing 1 to 10 of 13 entries | | | | | | Previous | 1 | 2 Next |
| Volver al menú | | | | | | | | |
| Cerrar la sesión | | | | | | | | |

E. Insertar un nuevo libro

Para agregar un nuevo libro, usaremos el modelo Libro.java, descrito en el apartado anterior, el servlet InsertarLibro.java y la vista insertLibro.jsp.

A través de la vista, recibiremos los parámetros que gestionará el servlet. El servlet y el modelo, trabajarán conjuntamente para añadir la información a la base de datos. Una vez introducidos los datos, seremos redireccionados a la vista listLibro.jsp, para que podamos comprobar que la inserción se ha realizado correctamente.

A continuación, pasamos a detallar el servlet:

```
package es.studium.LoginServlet;
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import es.studium.modelo.Libro;
/**
 * Servlet implementation class HazAlgo
 */
public class InsertarLibro extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public InsertarLibro()
    {
        super();
    }
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException
    {
        try
        {
            HttpSession session = request.getSession(false);
            synchronized(session)
```

```

        {
            String todo = request.getParameter("todo");
            if(todo==null)
            {
            }
            else if(todo.equals("add"))
            {
                Libro nuevoLibro = new Libro(
                    0,
                    (request.getParameter("tituloLibro")),
                    Double.parseDouble(request.getParameter("precioLibro")),
                    Integer.parseInt(request.getParameter("cantidadLibro")),
                    Integer.parseInt(request.getParameter("idEditorialFK")),
                    Integer.parseInt(request.getParameter("idAutorFK")));

                if(nuevoLibro.insertarLibro(nuevoLibro))
                {

                    response.sendRedirect("/PracticaLibrosServlets/muestraLibro");
                }
                else
                {

                }

            }

        }

    }catch(Exception E){
        System.out.println(E.getMessage());
    }

}
/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException

```

```

    {
        // TODO Auto-generated method stub
        doPost(request, response);
    }
}

```

La vista que se encargará de la recepción de los datos a insertar es insertLibro.jsp:

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page session="true" import="java.util.*, es.studium.modelo.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//ES"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insercion de Libros</title>
<link rel="stylesheet" type="text/css" href=" ../css/styleInsert.css">
</head>
<body>
    <h1>Libros MVC: Inserción de Libros</h1>
    <hr />
    <form name="insertarLib" action=" ../insertarLibro" method = "POST">
    <input type="hidden" id="todo" name="todo" value="add">
    <div class="form1">
        <label for="tituloLibro" id="tituloLibro1">tituloLibro<br/></label>
        <label for="precioLibro" id="precioLibro1">precioLibro<br/></label>
        <label for="cantidadLibro" id="cantidadLibro1">cantidadLibro<br/></label>
        <label for="idEditorialFK" id="idEditorialFK1">idEditorialFK<br/></label>
        <label for="idAutorFK" id="idAutorFK1">idAutorFK<br/></label>
    </div>
    <div class="form2">
        <input name="tituloLibro" id="tituloLibro"/><br/>
        <input name="precioLibro" id="precioLibro"/><br/>
        <input name="cantidadLibro" id="cantidadLibro"/><br/>
        <select name="idEditorialFK">
            <%
                Editorial editorial = new Editorial();
                List <Editorial> editoriales = editorial.listarEditorial();
                session.setAttribute("editoriales", editoriales);

                List<Editorial>          listaEditoriales          =          (List<Editorial>)
session.getAttribute("editoriales");
            %>

```

```

        for (Editorial editorial1 : listaEditoriales) {

            // Scriptlet 1: Carg los libros en el SELECT

            out.println("<option value='" + editorial1.getIdEditorial() +
">");

            out.println(editorial1.getIdEditorial() + " | " +
editorial1.getNombreEditorial());

            out.println("</option>");

        }
        %>
    </select></br> <select name="idAutorFK">
        <%

            Autor autor = new Autor();
            List <Autor> autores = autor.listarAutor();
            session.setAttribute("autores", autores);
            List<Autor> listaAutores = (List<Autor>)
session.getAttribute("autores");
            for (Autor autor1 : listaAutores) {

                // Scriptlet 1: Carg los libros en el SELECT

                out.println("<option value='" + autor1.getIdAutor() + "'>");
                out.println(autor1.getIdAutor()+ " | " + autor1.getNombreAutor() + "
| " + autor1.getApellido1Autor() + " | " + autor1.getApellido2Autor() );
                out.println("</option>");

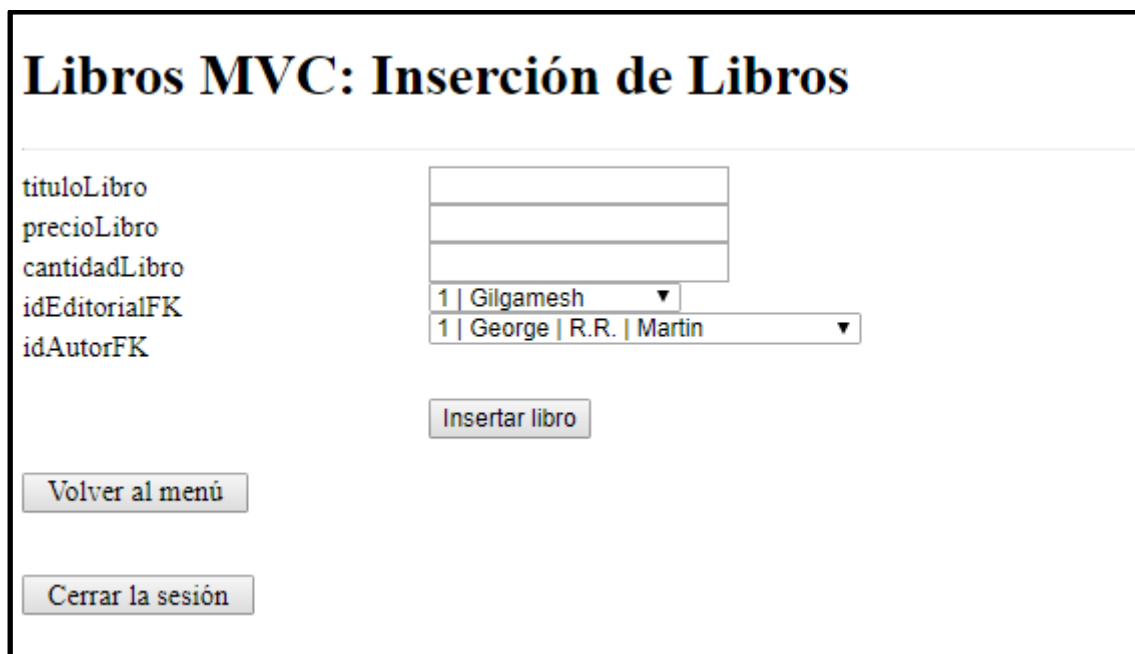
            }

            %>
        </select>

    </div>
    <br><br>
    <input id="submitButton" type="submit" value="Insertar libro">
    </form>
    <br />
    <a class="button" href="../menu.jsp">Volver al menú</a>
    <br><br>
    <a class="button" href="../logout">Cerrar la sesión</a>
</body>
</html>

```


La vista que recoge los datos (insertLibro.jsp) muestra este aspecto:



Libros MVC: Inserción de Libros

tituloLibro
precioLibro
cantidadLibro
idEditorialFK
idAutorFK

1 | Gilgamesh
1 | George | R.R. | Martin

Insertar libro

Volver al menú

Cerrar la sesión

La vista que se encarga de mostrar los libros, así como sus respectivos servlet y modelo ha sido explicada con anterioridad en el apartado D. Por ello, no repetiremos la información de manera innecesaria.

F. Modificar libro

Para la modificación de los libros, al igual que en el caso anterior, seremos redireccionados a una vista que recogerá la información necesaria para la modificación. Posteriormente, el servlet y el modelo, trabajarán de forma conjunta para realizar el cambio en la base de datos. Una vez realizado éste, seremos redirigidos a la vista encargada de mostrar el listado de libros de la base de datos para que podamos comprobar que la modificación se ha realizado correctamente.

El modelo se ha descrito con anterioridad y corresponde a Libro.java (véase apartado D). Pasamos, pues a desgranar el servlet encargado de realizar las oportunas modificaciones:

```
package es.studium.LoginServlet;
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import es.studium.modelo.Libro;
/**
 * Servlet implementation class HazAlgo
```

```
*/
public class ModificarLibro extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public ModificarLibro()
    {
        super();
        // TODO Auto-generated constructor stub
    }
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException
    {
        try
        {
            HttpSession session = request.getSession(false);
            synchronized(session)
            {

                String todo = request.getParameter("todo");
                if(todo==null)
                {

                }
                else if(todo.equals("modify"))
                {

                    Libro nuevoLibro = new Libro(

Integer.parseInt(request.getParameter("idLibro")),
                        (request.getParameter("tituloLibro")),

Double.parseDouble(request.getParameter("precioLibro")),

Integer.parseInt(request.getParameter("cantidadLibro")),

Integer.parseInt(request.getParameter("idEditorialFK")),

Integer.parseInt(request.getParameter("idAutorFK")));

                    if(nuevoLibro.modificarLibro(nuevoLibro))
```

```

        {

            response.sendRedirect("/PracticaLibrosServlets/muestraLibro");
        }
        else
        {

        }

    }

}

} catch (Exception E) {
    System.out.println(E.getMessage());
}

}
/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException
{
    // TODO Auto-generated method stub
    doPost(request, response);
}
}

```

La vista encargada de recoger los datos a modificar en nuestra base de datos es modificarLibro.jsp y está formada por el siguiente código:

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page session="true" import="java.util.*, es.studium.modelo.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//ES"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insercion de Libros</title>
<link rel="stylesheet" type="text/css" href=" ../css/styleInsert.css">
</head>
<body>
    <h1>Libros MVC: Modificación de Libros</h1>

```

```

<hr />
<form name="modificarLib" action="../modificarLibro" method="POST">
    <input type="hidden" id="todo" name="todo" value="modify">
    <div class="form1">
        <label for="idLibro" id="idLibro1">idLibro<br /></label> <label
            for="tituloLibro" id="tituloLibro1">tituloLibro<br
/></label> <label
            for="precioLibro" id="precioLibro1">precioLibro<br
/></label> <label
            for="cantidadLibro" id="cantidadLibro1">cantidadLibro<br
/></label>
        <label for="idEditorialFK"
id="idEditorialFK1">idEditorialFK<br /></label>
        <label for="idAutorFK" id="idAutorFK1">idAutorFK<br /></label>
    </div>
    <div class="form2">
        <input name="idLibro" id="idLibro" /><br /> <input
            name="tituloLibro" id="tituloLibro" /><br /> <input
            name="precioLibro" id="precioLibro" /><br /> <input
            name="cantidadLibro" id="cantidadLibro" /><br /> <select
            name="idEditorialFK">
            <%
                Editorial editorial = new Editorial();
                List <Editorial> editoriales =
editorial.listarEditorial();
                session.setAttribute("editoriales", editoriales);

                List<Editorial> listaEditoriales = (List<Editorial>)
session.getAttribute("editoriales");
                for (Editorial editorial1 : listaEditoriales) {

                    // Scriplet 1: Carg los libros en el SELECT

                    out.println("<option value='"
editorial1.getIdEditorial() + "'>");
                    out.println(editorial1.getIdEditorial() + " | " +
editorial1.getNombreEditorial());
                    out.println("</option>");
                }
            <%>
        </select></br> <select name="idAutorFK">
            <%
                //TODO Muestra los elementos del carrito
                Autor autor = new Autor();
                List <Autor> autores = autor.listarAutor();
                session.setAttribute("autores", autores);
                List<Autor> listaAutores = (List<Autor>)
session.getAttribute("autores");
            <%>
        </select>
    </div>
</form>

```

```

        for (Autor autor1 : listaAutores) {

            // Scriptlet 1: Carg los libros en el SELECT

            out.println("<option value='" + autor1.getIdAutor() +
"'">");

            out.println(autor1.getIdAutor()+ " | " +
autor1.getNombreAutor() + " | " + autor1.getApellido1Autor() + " | " +
autor1.getApellido2Autor() );
            out.println("</option>");

        }
    %>

</select>
</div>
<br/> <br /> <input id="submitButton" type="submit"
value="Realizar modificación">
</form>
<br />
<a class="button" href="../menu.jsp">Pulsa aquí para volver al menú</a>
<br />
<br />
<a class="button" href="../logout">Cerrar la sesión</a>
</body>
</html>

```

El aspecto visual que presenta la vista es el siguiente:

Libros MVC: Modificación de Libros

idLibro

tituloLibro

precioLibro

cantidadLibro

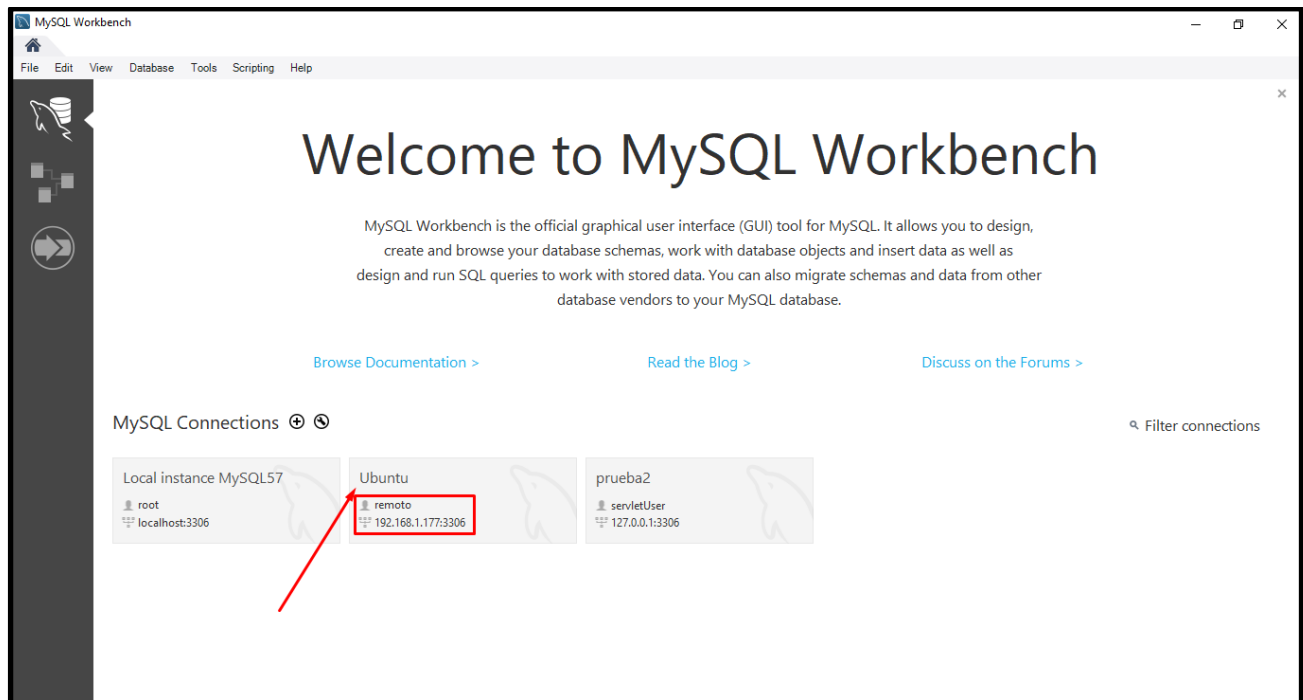
idEditorialFK

idAutorFK

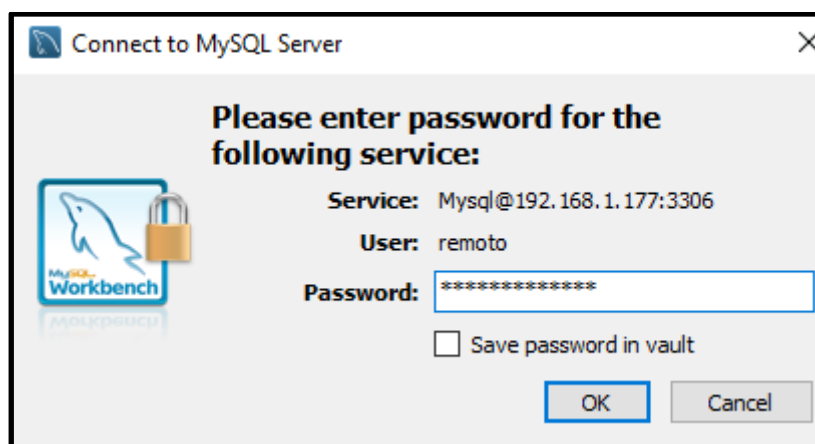
VI. Despliegue .WAR

Para acometer esta labor, es importante tener correctamente configurado nuestro servidor Ubuntu (que realizamos en prácticas anteriores). Es importante, para facilitarnos la labor, tener configurado el acceso al servidor a través de MySQL Workbench de forma remota.

Lo primero que haremos será en MySQL Workbench desde nuestro equipo físico y acceder remotamente al servidor, que previamente hemos puesto en funcionamiento.



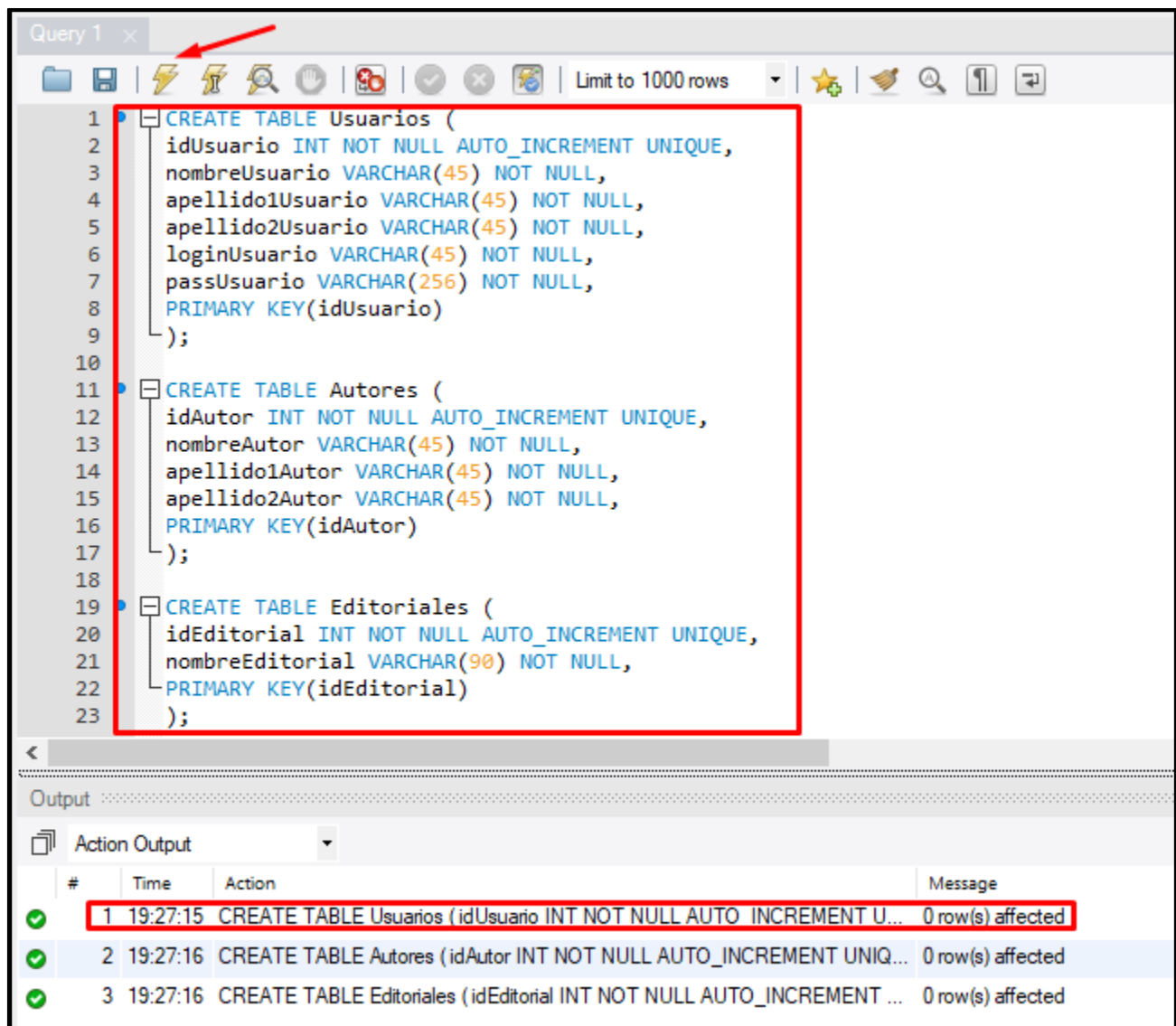
Introducimos nuestra contraseña de acceso remoto para hacer login.



El siguiente paso será, una vez dentro, borrar las tablas que conformaban la antigua BD y que por motivos de diseño se había quedado obsoleta.

| Action Output | | | | |
|---------------|----------|---|-------------------|--|
| # | Time | Action | Message | |
| ✓ 1 | 19:23:50 | DROP TABLE `tiendalibros`.`Libros` | 0 row(s) affected | |
| ✓ 2 | 19:23:54 | DROP TABLE `tiendalibros`.`Editoriales` | 0 row(s) affected | |
| ✓ 3 | 19:24:00 | DROP TABLE `tiendalibros`.`Autores` | 0 row(s) affected | |

El siguiente paso, consiste en reconstruir la base de datos a partir del fichero SQL que hemos creado con la nueva base de datos. En este paso, iremos añadiendo las tablas y realizaremos también los inserts que habíamos preparado con antelación. Si quisiéramos acceder a nuestro servidor Ubuntu para ejecutar esta operación y hubiéramos olvidado nuestra clave SUDO, consultar el apéndice A.



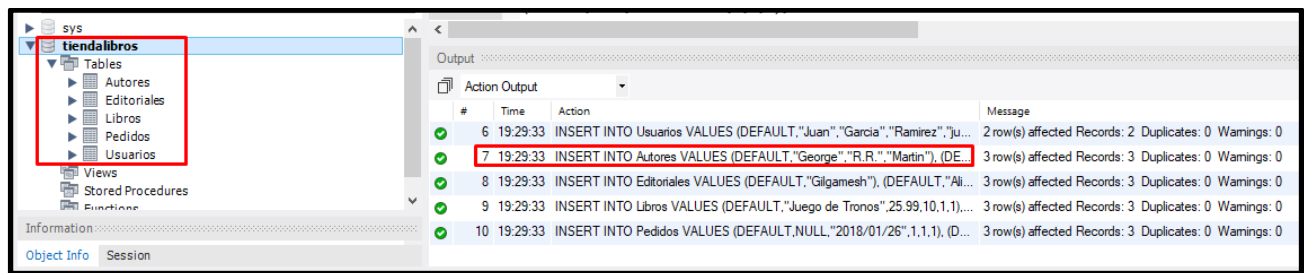
The screenshot shows a database management tool interface. At the top, there's a toolbar with various icons. Below it, a query editor displays three SQL queries for creating tables: 'Usuarios', 'Autores', and 'Editoriales'. The queries are highlighted with a red box. Below the query editor, there's an 'Output' section with a table showing the execution results of these queries. The first row of the output table is highlighted with a red box.

```
1 CREATE TABLE Usuarios (  
2   idUsuario INT NOT NULL AUTO_INCREMENT UNIQUE,  
3   nombreUsuario VARCHAR(45) NOT NULL,  
4   apellido1Usuario VARCHAR(45) NOT NULL,  
5   apellido2Usuario VARCHAR(45) NOT NULL,  
6   loginUsuario VARCHAR(45) NOT NULL,  
7   passUsuario VARCHAR(256) NOT NULL,  
8   PRIMARY KEY(idUsuario)  
9 );  
10  
11 CREATE TABLE Autores (  
12   idAutor INT NOT NULL AUTO_INCREMENT UNIQUE,  
13   nombreAutor VARCHAR(45) NOT NULL,  
14   apellido1Autor VARCHAR(45) NOT NULL,  
15   apellido2Autor VARCHAR(45) NOT NULL,  
16   PRIMARY KEY(idAutor)  
17 );  
18  
19 CREATE TABLE Editoriales (  
20   idEditorial INT NOT NULL AUTO_INCREMENT UNIQUE,  
21   nombreEditorial VARCHAR(90) NOT NULL,  
22   PRIMARY KEY(idEditorial)  
23 );
```

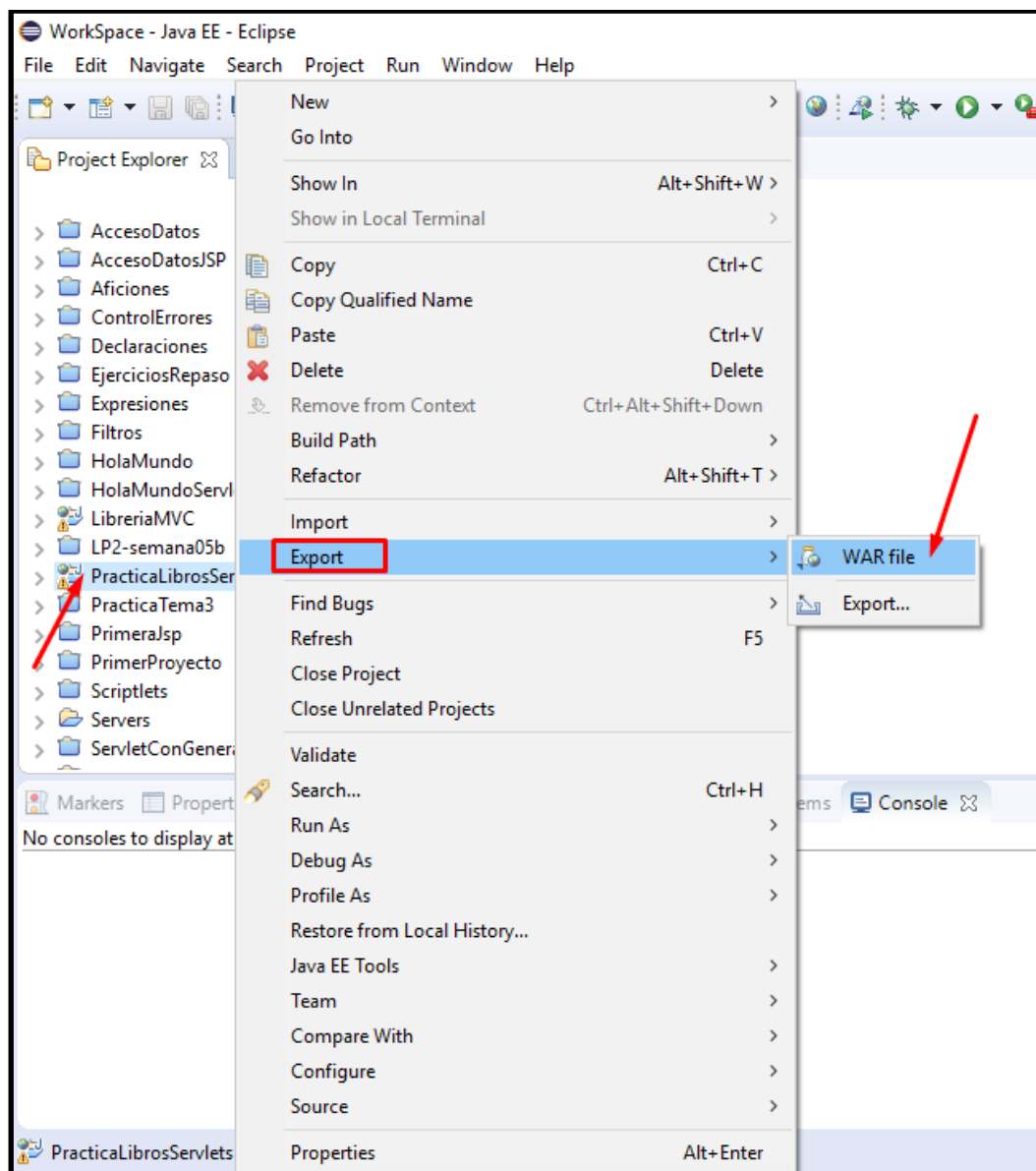
| Action Output | | | | |
|---------------|----------|---|-------------------|--|
| # | Time | Action | Message | |
| ✓ 1 | 19:27:15 | CREATE TABLE Usuarios (idUsuario INT NOT NULL AUTO_INCREMENT U... | 0 row(s) affected | |
| ✓ 2 | 19:27:16 | CREATE TABLE Autores (idAutor INT NOT NULL AUTO_INCREMENT UNIQ... | 0 row(s) affected | |
| ✓ 3 | 19:27:16 | CREATE TABLE Editoriales (idEditorial INT NOT NULL AUTO_INCREMENT ... | 0 row(s) affected | |

Realizaremos estos pasos con todas y cada una de las tablas.

Como se aprecia en la siguiente captura, hemos montado nuestra base de datos, con la estructura esperada y los inserts no presentan errores ni complicaciones.

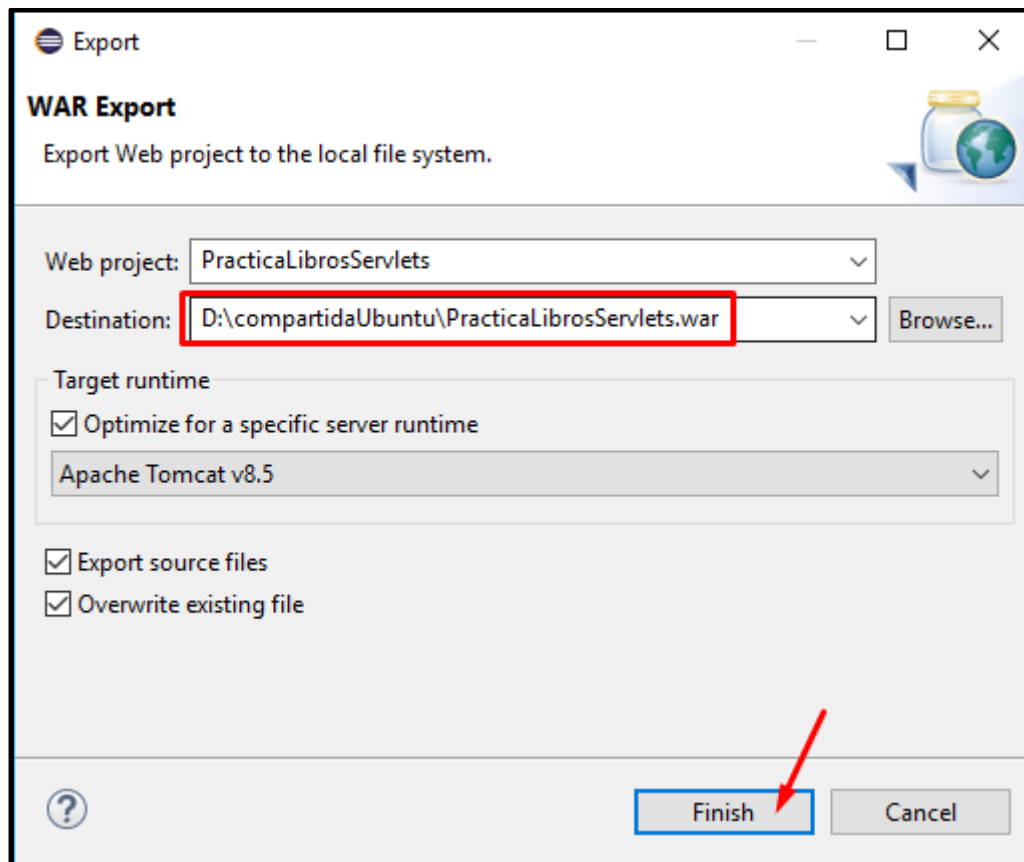


Pasamos ahora a nuestro Eclipse, para realizar la exportación del proyecto como fichero WAR:

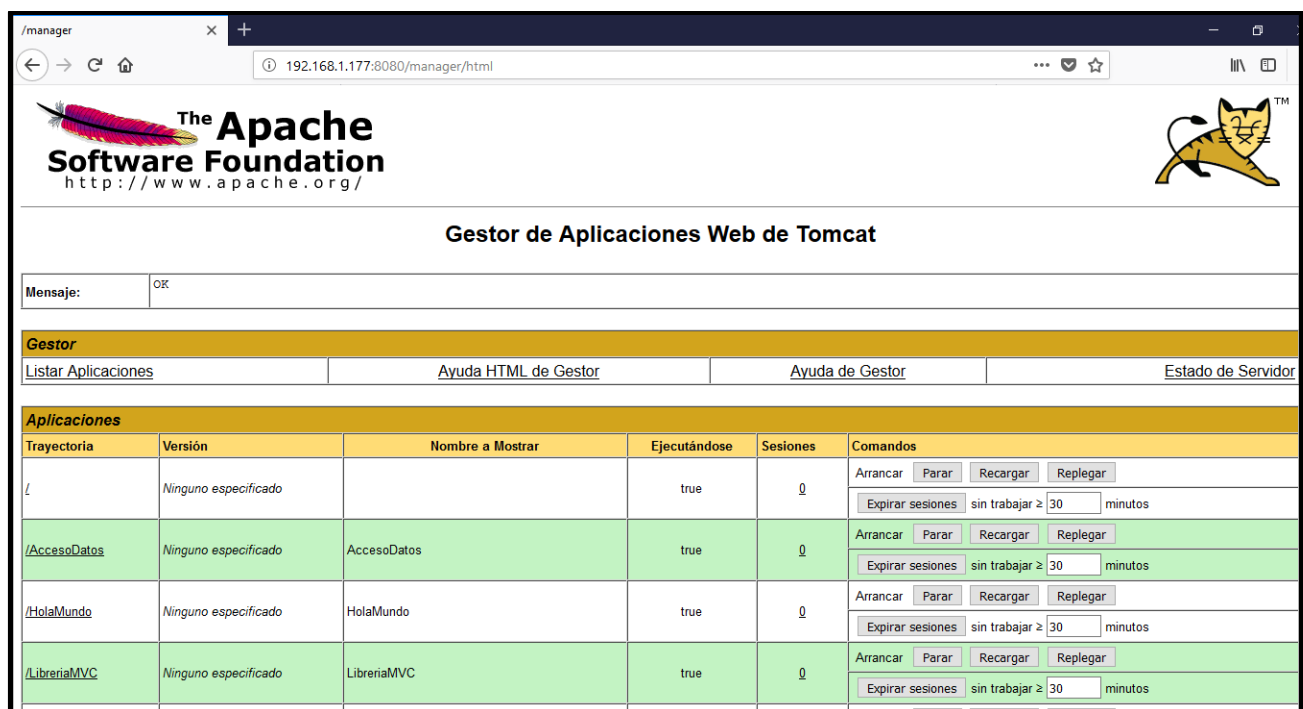


Haciendo clic derecho sobre el nombre del proyecto, luego iremos a exportar y finalmente a fichero WAR.

Tras esto seleccionaremos el destino en el que guardaremos el fichero.



El siguiente paso, será acceder remotamente a nuestro servidor Ubuntu y acceder al manager webapp de Tomcat. Debemos introducir nuestro login y contraseña:



Aquí realizaremos el despliegue de nuestra aplicación como hicimos en prácticas anteriores:

Desplegar

Desplegar directorio o archivo WAR localizado en servidor

Trayectoria de Contexto (opcional):
URL de archivo de Configuración XML:
URL de WAR o Directorio:

Archivo WAR a desplegar

Seleccione archivo WAR a cargar **PracticaLibrosServlets.war**

Tras este paso, podemos comprobar que nuestra aplicación web ha sido desplegada de la siguiente forma:

| Gestor de Aplicaciones Web de Tomcat | | | | | | |
|--------------------------------------|----------------------|--------------------------------------|--------------|---------------------------------|--|-------------------------------|
| Mensaje: | OK | | | | | |
| | | | | | | |
| Gestor | | | | | | |
| Listar Aplicaciones | | Ayuda HTML de Gestor | | Ayuda de Gestor | | Estado de Ser |
| | | | | | | |
| Aplicaciones | | | | | | |
| Trayectoria | Versión | Nombre a Mostrar | Ejecutándose | Sesiones | Comandos | |
| / | Ninguno especificado | | true | 0 | Arrancar <input type="button" value="Parar"/> <input type="button" value="Recargar"/> <input type="button" value="Replegar"/> <input type="button" value="Expirar sesiones"/> sin trabajar ≥ 30 minutos | |
| /AccesoDatos | Ninguno especificado | AccesoDatos | true | 0 | Arrancar <input type="button" value="Parar"/> <input type="button" value="Recargar"/> <input type="button" value="Replegar"/> <input type="button" value="Expirar sesiones"/> sin trabajar ≥ 30 minutos | |
| /HolaMundo | Ninguno especificado | HolaMundo | true | 0 | Arrancar <input type="button" value="Parar"/> <input type="button" value="Recargar"/> <input type="button" value="Replegar"/> <input type="button" value="Expirar sesiones"/> sin trabajar ≥ 30 minutos | |
| /LibreriaMVC | Ninguno especificado | LibreriaMVC | true | 0 | Arrancar <input type="button" value="Parar"/> <input type="button" value="Recargar"/> <input type="button" value="Replegar"/> <input type="button" value="Expirar sesiones"/> sin trabajar ≥ 30 minutos | |
| /PracticalLibrosServlets | Ninguno especificado | PracticalLibrosServlets | true | 0 | Arrancar <input type="button" value="Parar"/> <input type="button" value="Recargar"/> <input type="button" value="Replegar"/> <input type="button" value="Expirar sesiones"/> sin trabajar ≥ 30 minutos | |

Si la ejecutamos en un navegador, está operativa y funcional:

Login

192.168.1.177:8080/PracticaLibrosServlets/

Acceso

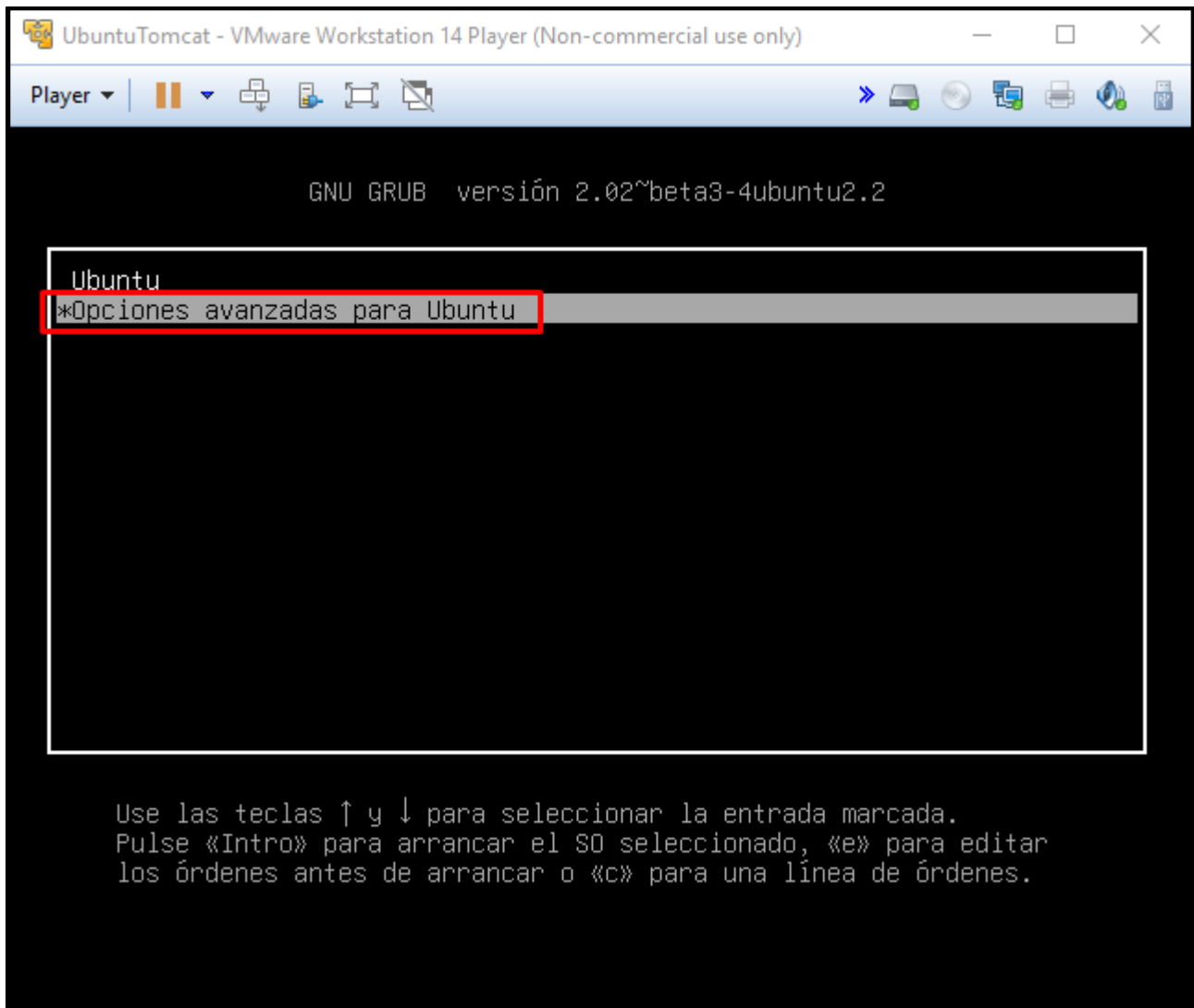
Username

Password

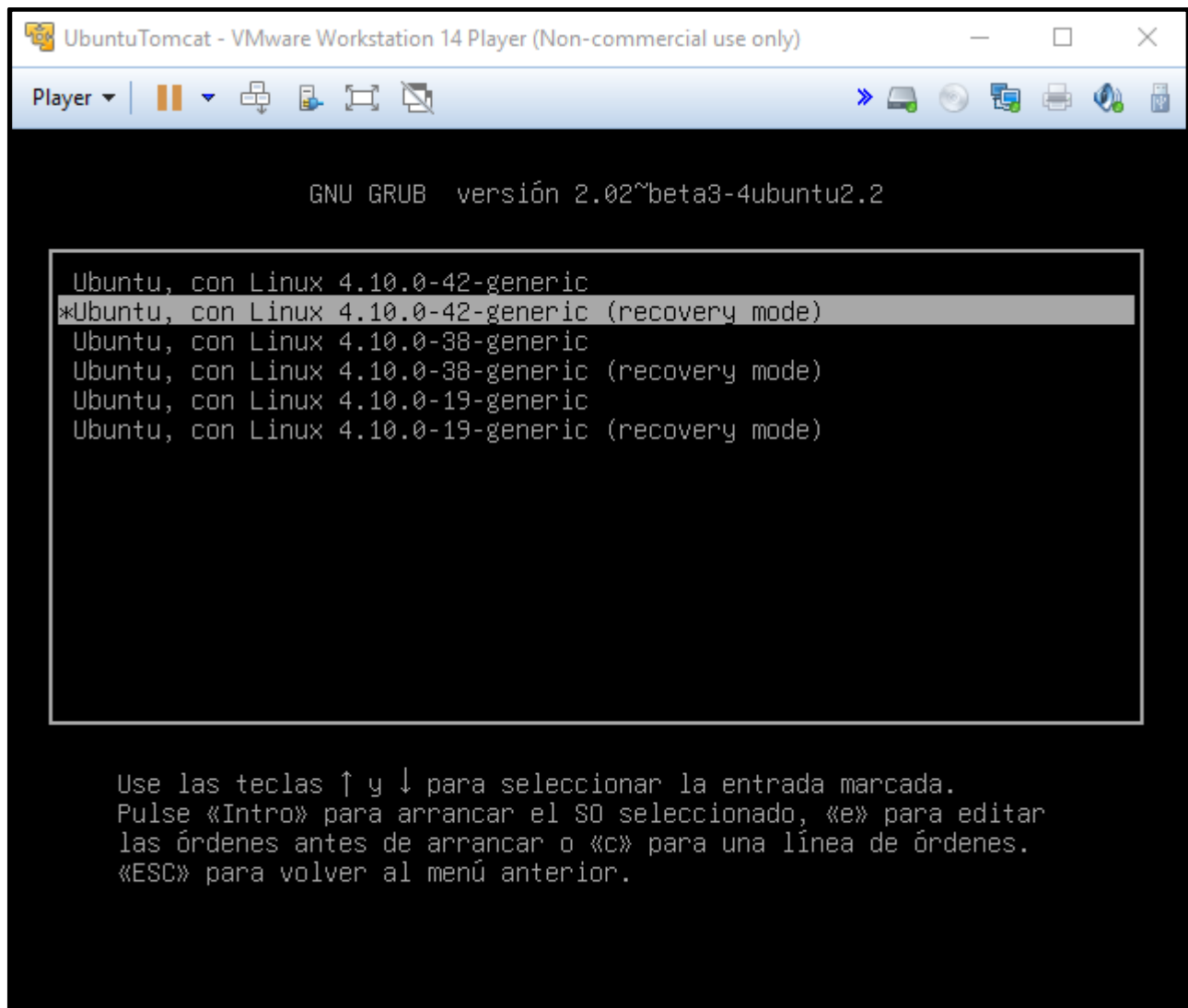
VII. Apéndice

A. Recuperar pass SUDO

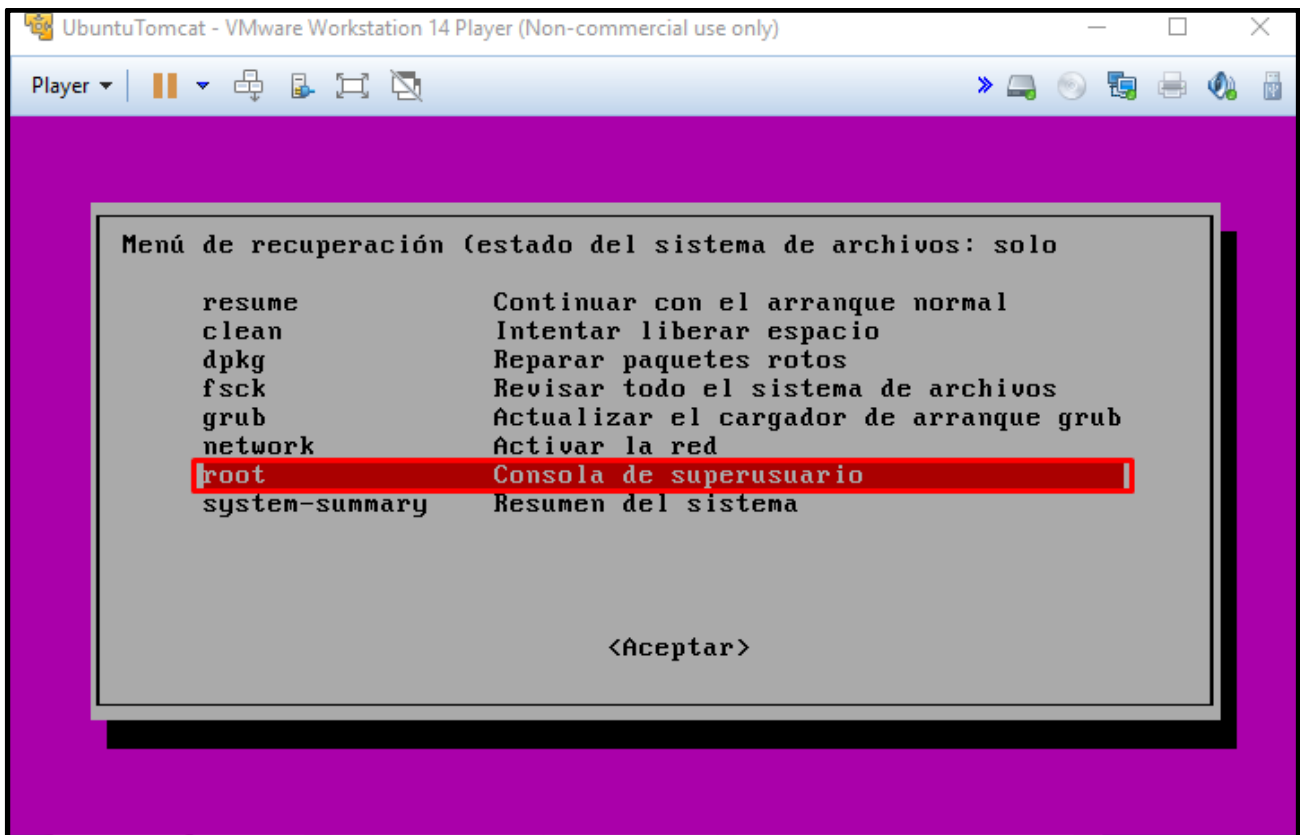
Al iniciar la máquina virtual, accederemos a las opciones avanzadas de Ubuntu.



y cuando nos sale el Grub, seleccionamos arrancar en modo recovery (recuperación). Y en el menú del Grub que nos aparece, elige recovery mode (recuperación).

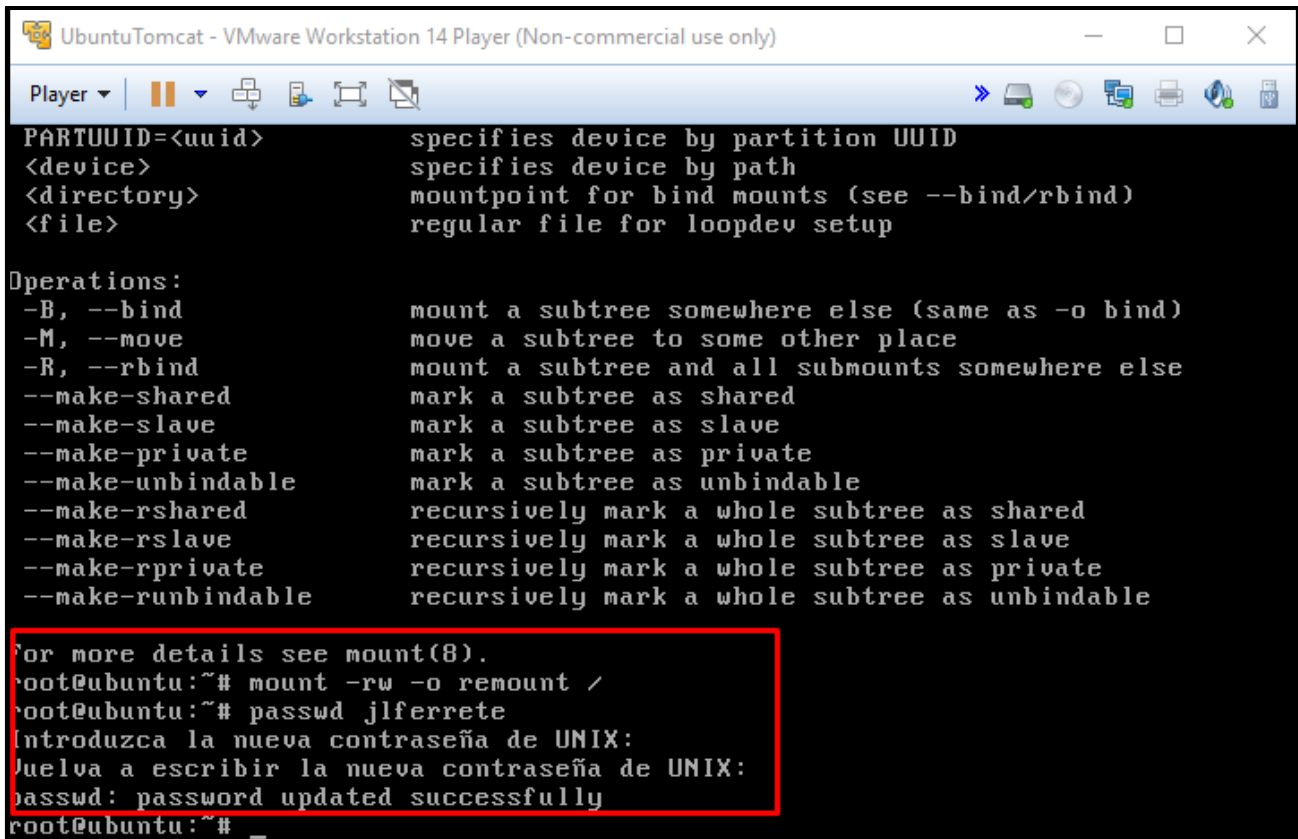


En el submenú que aparece elige "netroot" ("root" en las nuevas versiones), para abrir una terminal como superusuario sin contraseña.



La partición raíz "/", por defecto, se monta con protección contra escritura, por lo que debemos de volver a montarla con permisos de lectura y escritura (rw) para poder modificar el sistema:

```
Mount -rw -o remount /
```



```
UbuntuTomcat - VMware Workstation 14 Player (Non-commercial use only)
Player ▾ | [Icons]
PARTUUID=<uuid>      specifies device by partition UUID
<device>             specifies device by path
<directory>          mountpoint for bind mounts (see --bind/rbind)
<file>               regular file for loopdev setup

Operations:
-B, --bind            mount a subtree somewhere else (same as -o bind)
-M, --move            move a subtree to some other place
-R, --rbind           mount a subtree and all submounts somewhere else
--make-shared         mark a subtree as shared
--make-slave          mark a subtree as slave
--make-private        mark a subtree as private
--make-unbindable     mark a subtree as unbindable
--make-rshared        recursively mark a whole subtree as shared
--make-rslave         recursively mark a whole subtree as slave
--make-rprivate       recursively mark a whole subtree as private
--make-runbindable    recursively mark a whole subtree as unbindable

For more details see mount(8).
root@ubuntu:~# mount -rw -o remount /
root@ubuntu:~# passwd jlferrete
Introduzca la nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: password updated successfully
root@ubuntu:~# _
```

Ahora cambia tu contraseña ejecutando el comando:

```
passwd tu_nombre_usuario
```

Como podemos ver en la captura anterior, pulsa Enter y escribe tu nueva contraseña, no se verá nada en la pantalla, no te preocupes, es por seguridad, pero en realidad está escribiendo.

Pulsa Enter y te pedirá que la repitas. La escribes de nuevo y vuelve a pulsar Enter.

Apaga el sistema con el comando:

```
shutdown -r 0
```

Pulsa Enter y arranca el ordenador de forma normal, con tu nombre de usuario y la nueva contraseña.

VIII. Bibliografía

GrupoStudium. 2018. Apuntes de la asignatura. Sevilla : s.n., 2018.