

## **Proyecto ISDCM**

En las sesiones de prácticas de esta asignatura se va a realizar un proyecto que consiste en desarrollar una aplicación web para gestionar vídeos. Este proyecto se divide en varias entregas. El proyecto completo y sus entregas se describen brevemente a continuación.

En la **primera entrega** se desarrollará una aplicación web para la gestión de usuarios y vídeos. Esta entrega está descrita en este documento. El resto de entregas se publicará próximamente en el Racó.

En la **segunda entrega** se integrará la aplicación web con un servicio web basado en REST que implementará diversas búsquedas y contabilizará el número de veces que se hace *streaming* de los vídeos del sistema. En esta entrega también se tiene que desarrollar la funcionalidad de *streaming*.

En la **tercera entrega** se aplicarán distintas técnicas de seguridad a las aplicaciones desarrolladas.

Cada entrega tendrá una fecha límite en el Racó, apartado Prácticas. La evaluación de cada entrega se hará en base a la rúbrica de corrección también publicada en el Racó.

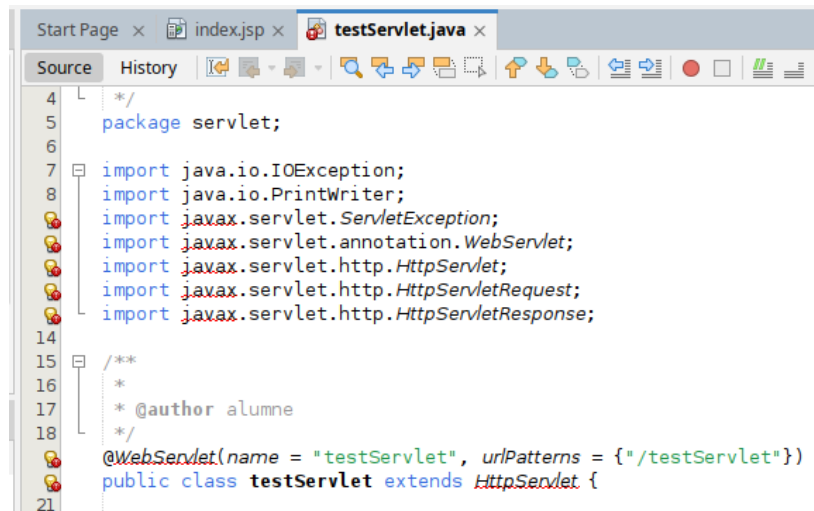
### **Entrega 1: Desarrollo de una aplicación web**

En esta primera entrega se va a desarrollar la parte correspondiente al registro de usuarios y vídeos y al listado de vídeos disponibles en el sistema. Las operaciones sobre vídeos sólo las podrán realizar los usuarios que hayan entrado correctamente en el sistema. La funcionalidad de búsqueda de vídeos se implementará en la **segunda entrega**.

El entorno de desarrollo del laboratorio que os proponemos está instalado en una máquina virtual Linux en las máquinas de laboratorio y también es accesible a través de Aula virtual (<https://aulavirtual.fib.upc.edu>). Esta máquina virtual se puede descargar de <https://softdocencia.fib.upc.edu/software/> (Es la que pone “Imatge virtual per assignatures GRAU-AD / MEI-ISDCM: Ubuntu-Netbeans24-V2.ova”). Este entorno de desarrollo es el IDE Netbeans 17 con Glassfish 6.5.2 y JDK 17. El resto del documento sigue este entorno de desarrollo, pero es posible utilizar otros, siempre que se mantenga la misma funcionalidad (o equivalente).

#### **Problema conocido curso 2024/2025:**

Las nuevas versiones de GlassFish Server utilizan el package **jakarta** y no **javax** como se hacía hasta ahora en los servlets. La solución es cambiar *javax* (*package generado automáticamente*) por *jakarta* y los errores de importación de clases desaparecen (ver Figuras 1 y 2).



```

4  L  */
5  package servlet;
6
7  import java.io.IOException;
8  import java.io.PrintWriter;
9  import javax.servlet.ServletException;
10 import javax.servlet.annotation.WebServlet;
11 import javax.servlet.http.HttpServlet;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
14
15 /**
16  *
17  * @author alumne
18  */
19 @WebServlet(name = "testServlet", urlPatterns = {"/testServlet"})
20 public class testServlet extends HttpServlet {
21

```

**Figura 1.** Error de importación de clases a la hora de crear un nuevo servlet



```

4  L  */
5  package servlet;
6
7  import java.io.IOException;
8  import java.io.PrintWriter;
9  import jakarta.servlet.ServletException;
10 import jakarta.servlet.annotation.WebServlet;
11 import jakarta.servlet.http.HttpServlet;
12 import jakarta.servlet.http.HttpServletRequest;
13 import jakarta.servlet.http.HttpServletResponse;
14
15 /**
16  *
17  * @author alumne
18  */
19 @WebServlet(name = "testServlet", urlPatterns = {"/testServlet"})
20 public class testServlet extends HttpServlet {
21

```

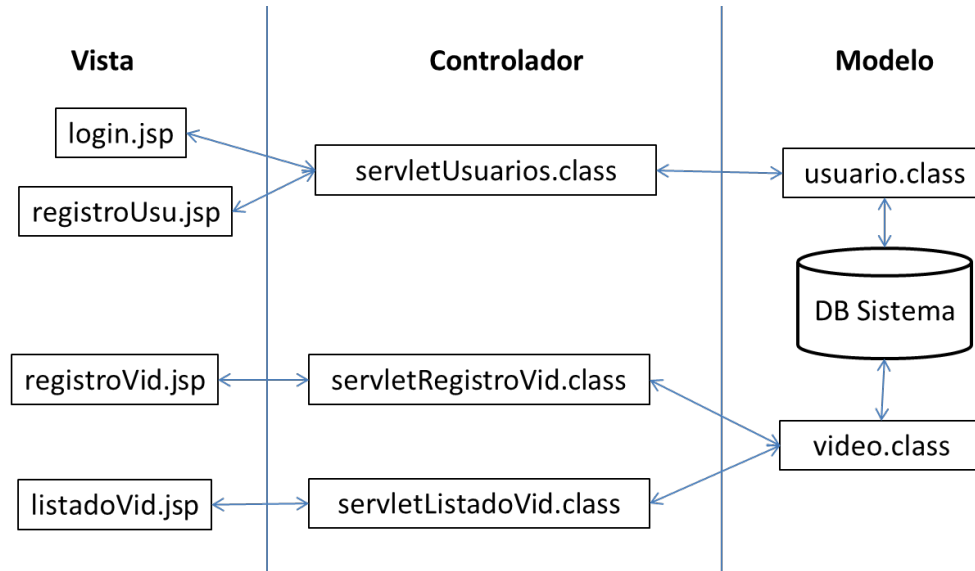
**Figura 2.** Cambio de *package* Java para resolver los errores

**Nota:** Para instalar el entorno en vuestras máquinas personales tenéis que utilizar la versión Apache Netbeans IDE 17 (URL descarga: <https://netbeans.apache.org/download/nb17/index.html>). También tenéis que instalar el JDK 17 de Oracle (URL descarga: <https://www.oracle.com/es/java/technologies/downloads/#java17>). Además, tenéis que instalar el servidor Glassfish Server 6.2.5 (URL descarga: <https://download.eclipse.org/ee4j/glassfish/glassfish-6.2.5.zip>). Esta combinación de versiones es la que hay (y funciona) en la máquina virtual de laboratorio. Se pueden utilizar otras versiones, pero tened en cuenta que hay muchas dependencias y que algunas combinaciones pueden fallar.

Una propuesta de esquema de la aplicación a desarrollar se muestra en la Figura 3. Se va a utilizar el patrón de diseño MVC (Modelo – Vista – Controlador) para separar la presentación de la aplicación (páginas jsp, Vista) del acceso a la base de datos (usuario.class, Modelo), teniendo como intermediario el modelo (servletUsuarios.class, Controlador). Se propone el uso de la base de datos Java DB que viene incluida con el IDE Netbeans que utilizaremos en el laboratorio. De nuevo, se pueden utilizar otras bases de datos relacionales (SQLite, MySQL, PostgreSQL, etc.), pero deberéis configurarlas para que funcionen en Netbeans con Glassfish.

La página login.jsp será para entrar en el sistema si el usuario ya está registrado y la página registroUsu.jsp permitirá registrar nuevos usuarios.

También se muestran las páginas y servlets de soporte para la funcionalidad de vídeos, registro y listado.



**Figura 3.** Propuesta de esquema de la aplicación

El usuario entrará en el sistema por la página login.jsp. En caso de no existir en el sistema, se le deberá dar la posibilidad de registrarse a través de la página registroUsu.jsp. Un ejemplo de presentación del registro de usuarios se muestra en la Figura 4.



Registro usuarios

Nombre:

Apellidos:

Correo electrónico:

Nombre usuario:

Contraseña:

Repetir contraseña:

**Figura 4.** Formulario de registro de usuarios

Los usuarios tendrán que introducir su nombre, apellidos, correo electrónico, nombre de usuario, contraseña y confirmación de contraseña en el formulario de la página inicial. Una vez hayan rellenado dichos campos, al clicar el botón **Registrar Usuario**, la información se envía al servlet controlador (servletUsuarios.class) que se encarga de procesarla y añadirla a la base de datos (DB Sistema), sólo si dicho usuario no está ya registrado. Por último, se muestra el resultado de la operación, indicando al usuario si se ha registrado correctamente o ya había algún usuario registrado con el mismo nombre de usuario. Se pueden añadir otras restricciones.

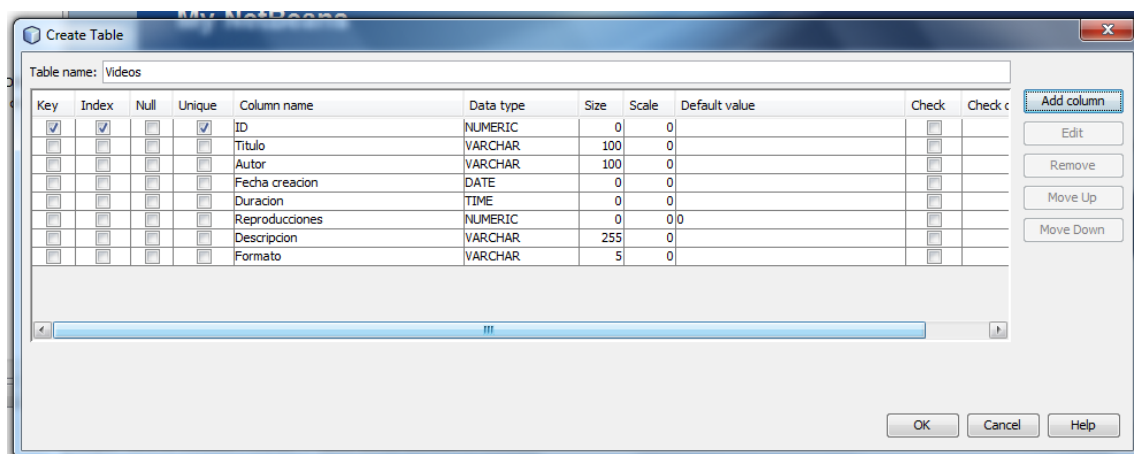
Es necesario configurar la aplicación web para que `login.jsp` sea vuestra página de inicio y no `index.html` o `index.jsp`. Además, es necesario implementar sesiones (con `HttpSession`) para asegurar que sólo los usuarios que han hecho login en el sistema pueden registrar y listar vídeos.

La funcionalidad de registro de vídeos tiene un funcionamiento similar al registro de usuarios. Se mostrará al usuario un formulario con los campos del vídeo que se van a almacenar en la base de datos. Para ello, debéis seguir el patrón MVC. Podéis añadir más páginas o servlets a la aplicación si lo creéis necesario. Como ya se ha dicho anteriormente, sólo los usuarios que hayan hecho login podrán registrar vídeos, por eso es necesario el uso de sesiones.

La información asociada a los vídeos debe ser, como mínimo, Identificador, Título, Autor, Fecha de creación, Duración, Número de reproducciones, Descripción y Formato.

Como parte de la aplicación, se requiere añadir el control de errores que creáis necesario a la aplicación web. Algunos ejemplos de errores pueden ser usuario repetido, vídeo repetido, errores de formato en los campos, control de campos vacíos, etc.

### Ejemplo de tabla de vídeos



**Figura 5.** Campos base de datos

### Ejemplo de datos en la tabla de vídeos

#	ID	TITULO	AUTOR	FECHA	DURACION	REPRODUCCIONES	DESCRIPCION	FORMATO
1		1 Big bunny	Peach Open Movie	2014-10-14	00:32:00		1 Historia de big bunny	ogg
2		2 Mr. Bean and Nurse	Christine	2014-10-14	05:38:00		0 Mr. Bean va al médico	mp4

**Figura 6.** Valores dentro de la base de datos

**Nota:** La base de datos de las Figuras 5 y 6 está incluida en Netbeans (Java DB). Esta base de datos no soporta opciones avanzadas de SQL.

**Forma de entrega**

Se debe entregar un fichero comprimido con la aplicación web desarrollada (ficheros html, jsp y servlets), sin los ficheros .jar de las librerías soporte. Además, se tiene que entregar un documento en el que se describa brevemente qué control de errores se ha implementado y dónde y cualquier elemento extra que se haya desarrollado en la práctica (uso de hojas de estilo, *frameworks*, etc.). Utilizad la plantilla que se os proporciona para entregar dicho informe.

No es necesario entregar copia de la base de datos, ya que las pruebas de la aplicación se realizarán en las máquinas del laboratorio (o en vuestras máquinas personales).

**Ampliaciones propuestas a la Entrega 1**

Los alumnos pueden proponer ampliaciones propias a esta entrega. Además, se propone la siguiente ampliación a esta práctica:

- Incluir en la base de datos un campo con la ruta al fichero de vídeo (puede ser una ruta local en vuestra máquina o una URL externa).

## Anexo I: Creación de una aplicación web en Apache Netbeans 17

Este anexo muestra cómo crear una aplicación web en Apache Netbeans 17. Para crear el proyecto hay que seleccionar Java with Maven → Web Application y pulsar Next (ver Figura 7).

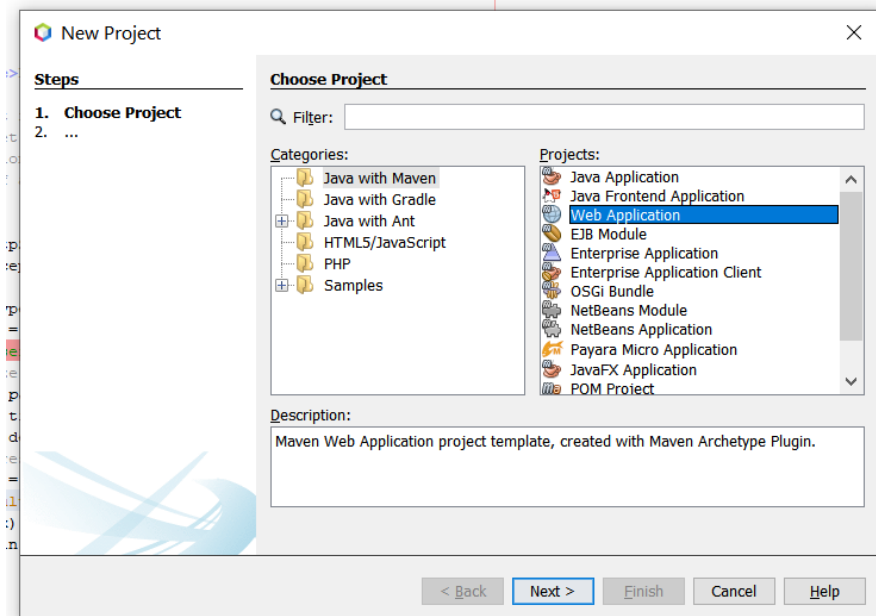


Figura 7. Creación de aplicación web

En la pantalla de parámetros del proyecto, tenéis que definir el nombre (webApp1 en la Figura 8). Se recomienda no utilizar espacios en blanco ni en las carpetas ni en el nombre del proyecto. También se puede definir el Group Id (en este caso es isdcm) y el Package donde se crearán las clases Java del proyecto. Una vez rellenada esta información, pulsar Next.

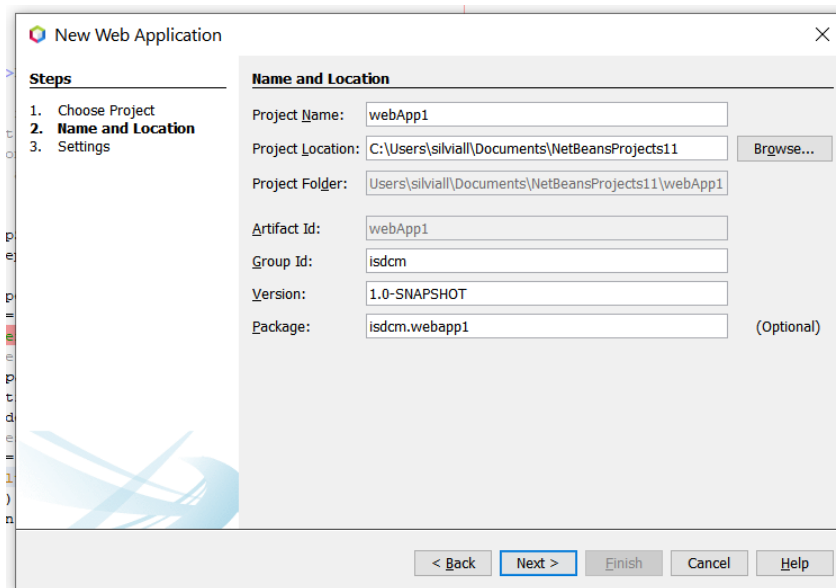
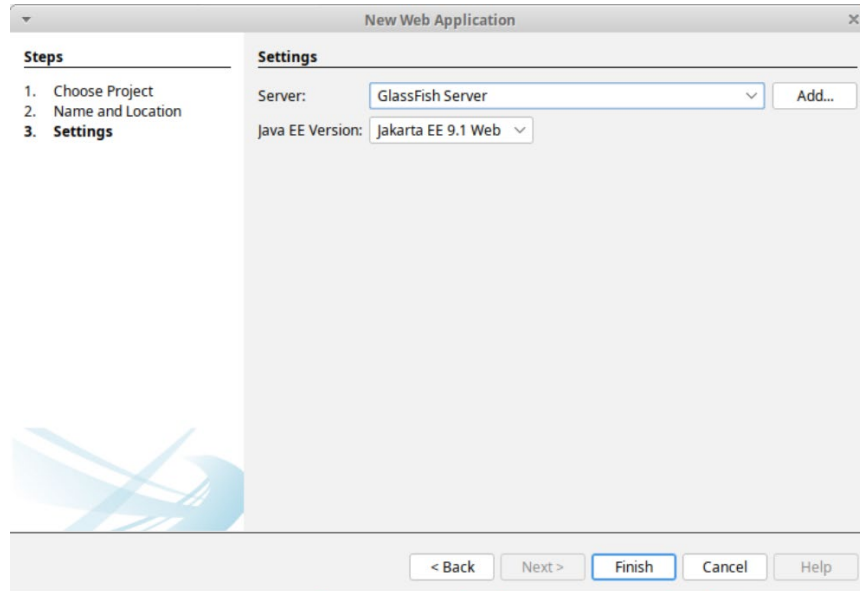
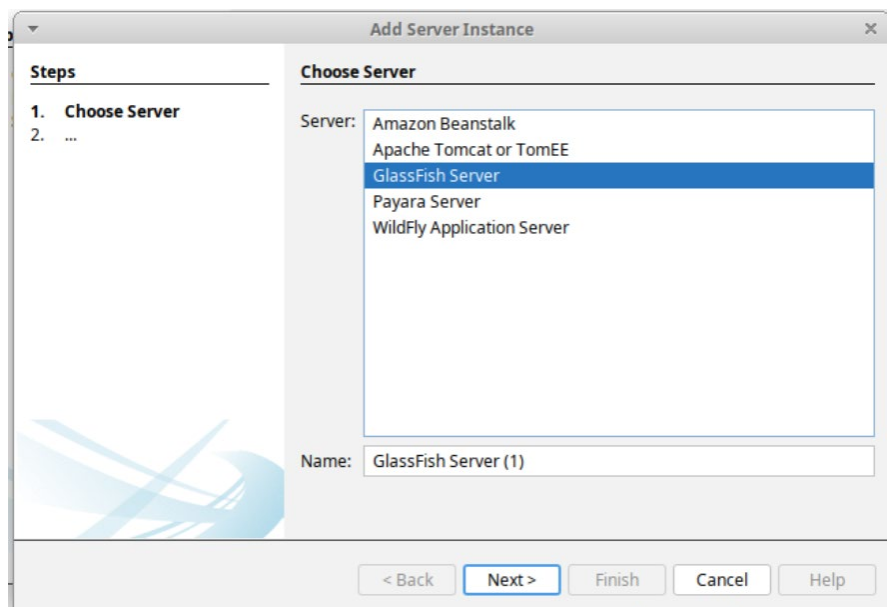


Figura 8. Parámetros de la aplicación web

El siguiente paso (Figura 9) es seleccionar el tipo de servidor que vamos a utilizar. En nuestro caso, utilizaremos Glassfish Server con la versión Jakarta EE 9.1 Web. Si la lista de servidores os sale vacía, tendréis que instalar el servidor (Figura 10). En el ordenador donde se han realizado estas capturas ya están instalados los servidores, así que no aparecen las pantallas de descarga del programa.

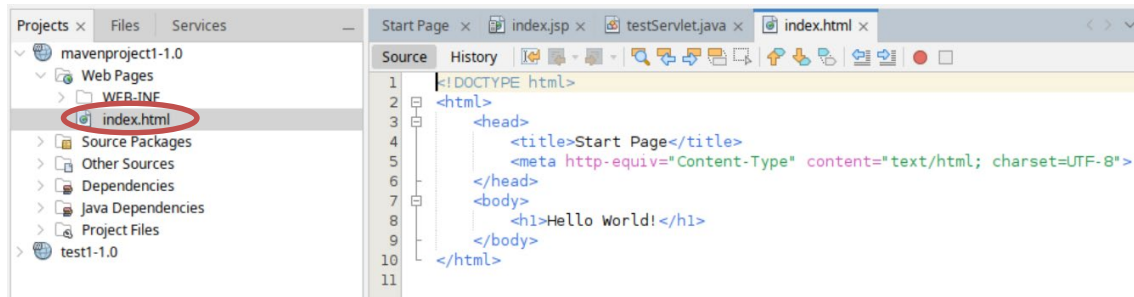


**Figura 9.** Selección del servidor

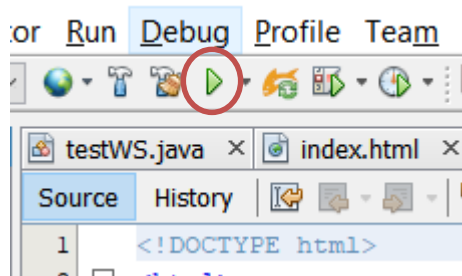
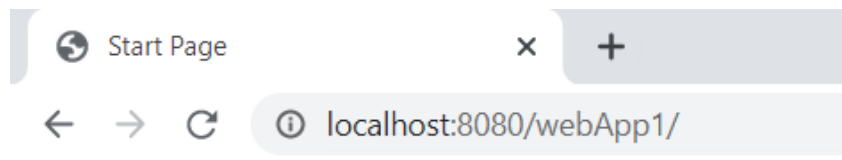


**Figura 10.** Añadir nuevo servidor

Para las dos primeras entregas del proyecto hay que utilizar Glassfish Server. Cuando ya esté instalado el servidor, pulsa Finish en la ventana de New Web Application (Figura 9), que creará la aplicación web. En la Figura 11 se puede ver la aplicación web que se acaba de crear, donde ya tenemos la página web index.html.

**Figura 11.** Nueva webApp1

Para ejecutar la aplicación se tiene que pulsar el botón seleccionado en la Figura 12. El resultado de la ejecución se puede ver en la Figura 13, donde la página `index.html` se ve en el navegador.

**Figura 12.** Puesta en marcha webApp1

# Hello World!

**Figura 13.** Ejecución de webApp1 y vista en el navegador



## Anexo 2: Soporte a la base de datos JavaDB

Este anexo describe cómo utilizar la JavaDB, una base de datos integrada con Netbeans y Glassfish. Sigue los siguientes pasos para poder acceder a una base de datos Java DB desde tu aplicación web.

Para iniciar la JavaDB, ve al panel Servicios y selecciona Start Server (dependiendo del idioma de Netbeans, el comando tendrá otro nombre). En la Figura 12, el servidor ya estaba iniciado, por eso no se puede seleccionar el comando. Después, selecciona Create Database y rellena los parámetros para crear la base de datos tal y como se muestra en la Figura 13. A continuación, conéctate con la base de datos como se puede ver en la Figura 14. La conexión jdbc:derby://localhost:1527/pr2 se ha activado. Ábrela y ve a la carpeta Tables. Selecciona la opción Execute Command (Figura 15). En el nuevo fichero que aparecerá (Figura 16), podéis escribir las sentencias SQL que necesitéis.

El nombre de la base de datos, usuario y password es pr2. Podéis modificarlo, pero recordad lo que habéis puesto porque lo utilizaremos en las siguientes prácticas.

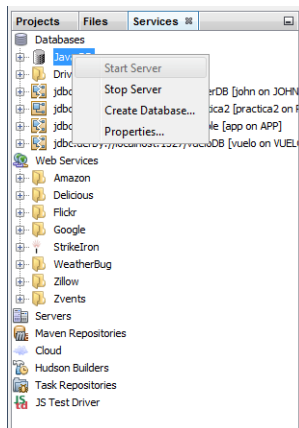


Figura 12. Iniciar base de datos

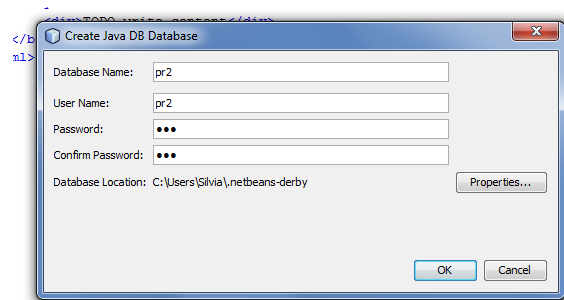


Figura 13. Crear base de datos

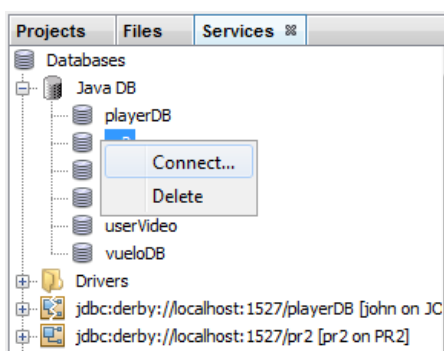


Figura 14. Conectar a la base de datos

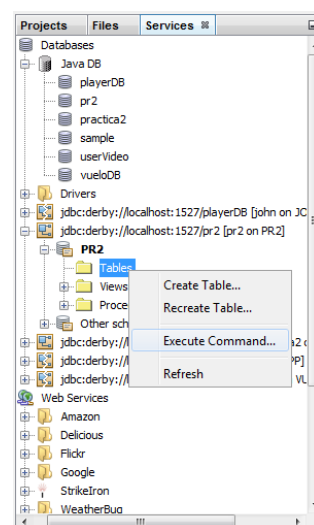
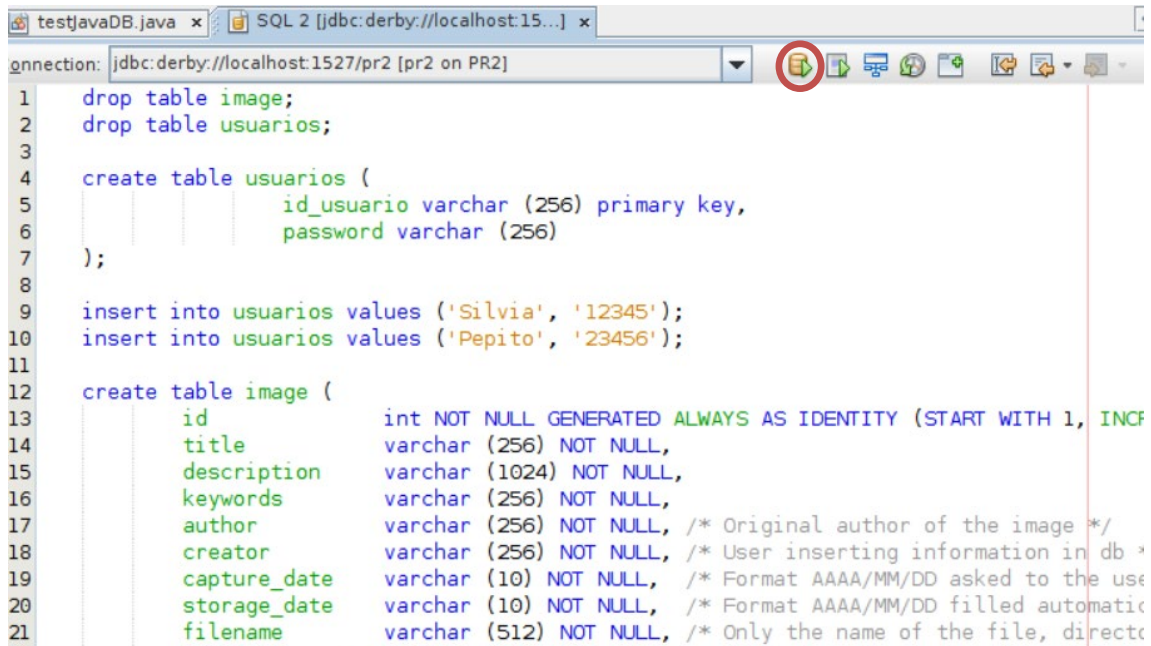


Figura 15. Ejecuta el comando SQL para crear la base de datos



```

1  drop table image;
2  drop table usuarios;
3
4  create table usuarios (
5      id_usuario varchar (256) primary key,
6      password varchar (256)
7  );
8
9  insert into usuarios values ('Silvia', '12345');
10 insert into usuarios values ('Pepito', '23456');
11
12 create table image (
13     id int NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCF
14     title varchar (256) NOT NULL,
15     description varchar (1024) NOT NULL,
16     keywords varchar (256) NOT NULL,
17     author varchar (256) NOT NULL, /* Original author of the image */
18     creator varchar (256) NOT NULL, /* User inserting information in db */
19     capture_date varchar (10) NOT NULL, /* Format AAAA/MM/DD asked to the use
20     storage_date varchar (10) NOT NULL, /* Format AAAA/MM/DD filled automatic
21     filename varchar (512) NOT NULL, /* Only the name of the file, direct
    
```

**Figura 16.** Ejemplo de fichero SQL. No contiene la misma información que se pide en el enunciado

La base de datos se ha creado. Ahora para poder utilizarla desde vuestros servlets y jsp's, tenéis que utilizar el código que se muestra en la Figura 17:

```

String query;
PreparedStatement statement;

Class.forName("org.apache.derby.jdbc.ClientDriver");

// create a database connection
connection = DriverManager.getConnection("jdbc:derby://localhost:1527/pr2;user=pr2;password=pr2");
    
```

**Figura 17.** Cargar driver y conectar con la base de datos

Una vez hecho esto, ya se pueden utilizar las sentencias SQL necesarias para implementar las funcionalidades requeridas en el enunciado.