Informe de prácticas de ISDCM

Entrega 1

Alumno/a: Javier Abella

Alumno/a: Ana Poveda



Contenido

[Introducción 2](#_Toc271714754)

[Decisiones de diseño 3](#_Toc618608126)

[Gestión Base de Datos 3](#_Toc1117291856)

[Estructura MVC 3](#_Toc1075550277)

[Modelo 3](#_Toc1900321619)

[Vistas 3](#_Toc933238284)

[Controlador 3](#_Toc1042094111)

[Control de errores: 4](#_Toc1739274935)

[Control de errores de ejecución en las redirecciones: 4](#_Toc180849109)

[Control valores de entrada en los formularios 4](#_Toc439184822)

[Estilos 4](#_Toc624576829)

[Repositorios de código consultados 5](#_Toc1996035516)

[Bibliografía consultada 5](#_Toc206449492)

# Introducción

La aplicación web creada en esta práctica consiste en iniciar sesión, registrar nuevos usuarios, registrar videos y visualizar una lista de los vídeos disponibles.

# Decisiones de diseño

## Gestión Base de Datos

Java DB utiliza la API JDBC estándar de Java para la gestión de bases de datos. Las clases que hemos utilizado para conectarnos a la base de datos son:

* **java.sql.Connection**: representa la conexión a la base de datos.
* **java.sql.Statement**: se usa para ejecutar consultas SQL sin parámetros.
* **java.sql.PreparedStatement**: se usa para ejecutar consultas SQL con parámetros.
* **java.sql.ResultSet**: contiene los resultados de una consulta SQL.
* **java.sql.DriverManager**: se usa para registrar y obtener la conexión con la base de datos

Para evitar conexiones directas con la base de datos, desde los controladores hemos añadido un paquete de archivos DAO donde se gestionan las llamadas a la base de datos.

## Estructura MVC

Para desarrollar la aplicación web hemos utilizado el patrón MVC (Modelo-Vista-Controlador). Esto se puede ver reflejado en el proyecto con todos los controladores *(servlets)* ubicados en un mismo paquete y lo mismo con los modelos. Las vistas están en la carpeta de webapp y al mismo nivel hay subcarpetas para los estilos aplicados.

Modelo

Hemos creado las clases **User.java** y **Video.java.**

Vistas

Hemos creado las siguientes vistas:

* **login.jsp:** vista para iniciar sesión.
* **registroUsu.jsp**: vista para registrar a un nuevo usuario.
* **registroVid.jsp:** vista para registrar un nuevo video.
* **listado.jsp:** vista que muestra todos los videos disponibles en el sistema.
* **error.jsp:** vista a la que dirigimos en caso de que haya un error en base de datos o se intente realizar una acción no válida.

### Controlador

Maneja las peticiones del usuario, interactúa con el modelo y decide qué vista mostrar. Hemos implementado los siguientes *servlets*:

* **servletListadoVid.java:** gestiona la acción de mostrar todos los videos del sistema.
* **servletRegistroVid.java:** gestiona las acciones de registrar un nuevo video.
* **servletUsuarios.java:** gestiona todas las acciones de iniciar sesión, registrar un nuevo usuario y cerrar sesión.

Control de errores:

Control de errores de ejecución en las redirecciones:

1. Gestionar errores de la Base de datos:

En los métodos para hacer login y para registrar a un nuevo usuario usamos la clase SQLException para gestionar excepciones de la base de datos. Esta clase nos permite obtener información acerca de los errores y avisos que se producen mientras accedemos a la base de datos. En caso de que se produzca un error, redirigimos al usuario s a una vista especifica, error.jsp.

#### Control de errores en los métodos doGet y doPost

Para evitar peticione son válidas, gestionamos tanto las acciones definidas por nosotros, por ejemplo, *register, login y logout*, y otras posibles acciones. De esta forma, i el usuario intenta acceder a servletUsuarios sin la acción correcta le redirijimos a la vista error.jsp.

#### Redirección con mensajes de errores persistentes

Para mantener los atributos de error y no perderlos con el método sendRedirect(), usamos la clase HttpSession en lugar de request.setAttibute() en el método loginUser() del servletUsuarios. A continuación, en la vista login.jsp recuperamos el error.

#### Controlar la expiración de la sesión

Verificamos si la sesión sigue activa antes de redirigir. Si un usuario deja su sesión abierta y vuelve después de un tiempo, la sesión puede expirar. En ese caso, se le redirigirá a la página de login.

### Control valores de entrada en los formularios

#### Validación de los campos del formulario de registro de usuario:

Para implementar la funcionalidad de registrar un usuario se han realizado las siguientes validaciones:

* Todos los campos del formulario deben ser informados.
* Los valores indicados en los campos de contraseña, *Contraseña* y *Repetir Contraseña*, deben coincidir.
* No puede existir un usuario con el mismo *Username*.
* La contraseña debe tener mínimo 8 caracteres, incluir letras, números y símbolos. Los símbolos especiales son: @, $, !, %, \*, ?, &.

#### Validación de los campos del formulario de inicio de sesión:

Para implementar la funcionalidad de iniciar sesión se han realizado las siguientes validaciones:

* Todos los campos del formulario deben ser informados.
* Las credenciales indicadas en el formulario deben coincidir con las de algún usuario registrado en la base de datos.

## Estilos

Hemos creado una carpeta css en la que hemos añadido ficheros Cascading Style Sheet. Hemos creado un .css por cada vista. De esta forma nos resulta mucho más sencillo localizar los estilos que queremos editar en una vista en concreto.

# Repositorios de código consultados

Lista de ejemplos de código consultados en github, gitlab, etc.

# Bibliografía consultada

<https://www.ciberaula.com/cursos/java/manejo_errores_excepciones_java.php>

<https://keepcoding.io/blog/errores-de-entrada-del-usuario/>

<https://developer.mozilla.org/es/docs/Learn_web_development/Extensions/Forms/Styling_web_forms>

<https://stackoverflow.com/questions/70096397/traditional-mvc-design-pattern>