Informe de prácticas de ISDCM

Entrega 2

Alumno/a: Albert Bausili Fernández

Alumno/a: Noa Yu Ventura Vila

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

FIB

Contenido

1. Introducción

Este informe contiene toda la documentación del código hecho para esta entrega. Primero de todo veremos un esquema para tener una idea de la arquitectura de la aplicación y ver cómo se comunican las distintas partes. Además, veremos una tabla con todos los endpoints disponibles y los métodos usados para cada uno. Luego habrá una breve y resumida explicación de las funcionalidades de la aplicación. También habrá un apartado de aclaraciones para justificar ciertas decisiones en la arquitectura de la aplicación y después un apartado para recalcar todos los extras que hemos hecho. Finalmente podremos ver los repositorios de código y bibliografía consultados.

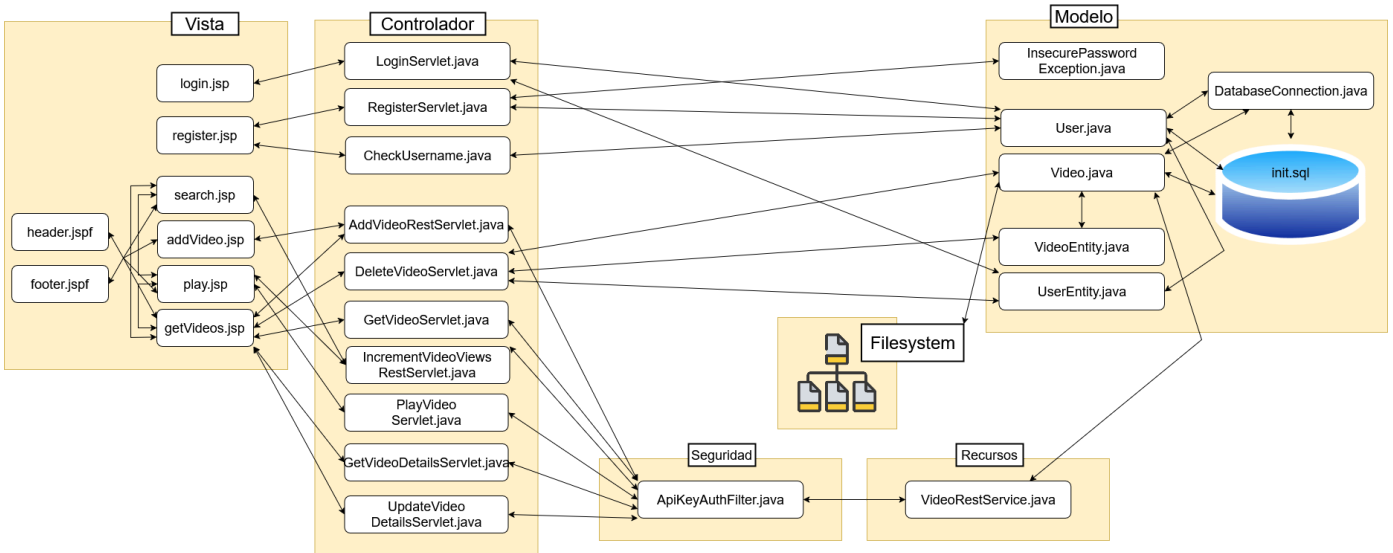
El propósito principal de esta entrega es modificar el código de la entrega 1 para que use un servicio REST y añadir la reproducción de los vídeos que hemos colgado.

2. Decisiones de diseño

2.1. Arquitectura de la Aplicación

Algunas vistas son las mismas que en la entrega anterior: el registro de usuarios, el inicio de sesión, la subida de un vídeo y el listado de vídeos. Además, hemos añadido dos más, el search para buscar y mostrar los vídeos filtrados y el play para reproducir el vídeo. Cabe destacar que el search no recoge todos los vídeos que hay y luego los filtra, sino que filtra antes de pedirlos, de modo que ahorra mucho tiempo y recursos de la aplicación. Además es escalable, ya que en el caso de haber millones y millones de vídeos solo coge unos cuantos a no ser que el usuario quiera más, haciendo la aplicación rápida y eficiente.

Hemos añadido un servicio REST por donde pasan algunas requests de los servlets, ya sea para subir un vídeo o para incrementar el número de visitas. En la siguiente figura se puede apreciar la lógica de la aplicación.



2.2. Explicación de Ficheros

Página	**Breve descripción**
login.jsp	Contiene la tag *style* y bootstrap para el diseño de la página. Comprueba que el nombre de usuario y la contraseña son correctos. Se puede acceder sin sesión.
register.jsp	Contiene la tag *style* y bootstrap para el diseño de la página. Tiene un script que en tiempo real llama a CheckUsername.java para ver si el nombre de usuario ya existe (mensaje de color verde indicando que el nombre de usuario está libre, o rojo con un error en caso contrario). Marca los campos que son requeridos y los que no, e impide registrarse si hay campos requeridos sin rellenar. Se puede acceder sin sesión.
addVideo.jsp	Contiene la tag *style* y bootstrap para el diseño de la página. Vista para colgar un nuevo vídeo después de darle al botón "Upload File" en la vista getVideos.jsp. Contiene un script que envía el vídeo y la información introducida por el usuario (Título y descripción) al servlet y automáticamente obtiene la duración del vídeo. Este servlet es el que luego hace la petición al servicio REST. Requiere sesión.
getVideos.jsp	Contiene la tag *style* y bootstrap para el diseño de la página. Comprueba que haya un usuario que se haya logueado para mostrar el icono de borrar video solo en los que el usuario haya subido. Cada video tiene un botón para reproducir mediante streaming de un servlet y al reproducirlo llama a otro servlet que aumenta el número de vistas en 1. Requiere sesión.
search.jsp	Muy parecido al getVideos.jsp con la peculiaridad que permite buscar imágenes según ciertos filtros establecidos por el usuario. Requiere sesión.
play.jsp	Reproducir los vídeos al darle al botón de *play*. Es el que incrementa el número de visitas del vídeo.
LoginServlet.java	Tiene las mismas comprobaciones de errores que en el JSP y llama a User.java para comprobar que el nombre de usuario y la contraseña son correctos. Si son correctos inicia la sesión del usuario.
RegisterServlet.java	Tiene las mismas comprobaciones de errores que en el JSP para ver si la contraseña es segura (en caso negativo lanza un InsecurePasswordException). Si los datos son correctos los envía a la capa modelo para guardarlos en la DB y redirige al usuario al login. No requiere sesión.
CheckUsername.java	Llama a User.java para comprobar que el nombre de usuario no existe aún (y si no impide proseguir con el registro). Se usa cada vez que el usuario deja de teclear durante un tiempo específico en el campo de Usuario en el formulario de registro. No requiere sesión.
AddVideoRestServlet.java	Crea un nuevo vídeo con las características que recibe con la request y llama al servicio REST para que éste lo añada a la base de datos y lo guarde

	también en el filesystem para que después se pueda recuperar. Requiere sesión.
DeleteVideoServlet.java	Cuando el servlet recibe la petición lo envía al servicio REST, y éste comprueba que el vídeo exista, que quien quiere borrarlo sea el autor de ese vídeo y que la operación haya sido exitosa. Requiere sesión.
GetVideoServlet.java	Devuelve los vídeos que están subidos en la plataforma según unos filtros establecidos por el usuario (y si no lo están, pues devuelve los que haya). Para hacerlo se lo pide al servicio REST. Requiere sesión.
IncrementVideo ViewsRestServlet.java	Cuando el servlet recibe la petición envía un POST para incrementar el número de visitas del vídeo que dice el parámetro en 1. Requiere sesión.
UserEntity.java	Clase usuario con sus atributos y métodos. Requiere sesión.
VideoEntity.java	Clase vídeo con sus atributos y métodos. Requiere sesión.
User.java	Gestiona todas las llamadas a la base de datos relacionadas con el usuario: registro, autenticación, hashear contraseña y comprobar si el nombre de usuario existe. Requiere sesión.
Video.java	Gestiona todas las llamadas a la base de datos relacionadas con los vídeos: añadir uno nuevo, obtener todos los vídeos, incrementar las visitas en 1 de un vídeo en concreto, obtener un vídeo por ID y borrar vídeos. Requiere sesión.
DatabaseConnection.java	Crea la conexión con la base de datos. Requiere sesión.
InsecurePassword Exception.java	La comprobación de que la contraseña que quiere crear el usuario en la vista de registro es segura solamente está implementada en el servlet de registrar usuario (no ha dado tiempo a hacerlo en el JSP, pero en un futuro lo implementaremos). Cuando la contraseña no es segura se lanza esta excepción explicándole al usuario los requerimientos para que la contraseña sea segura. Requiere sesión.

2.3. Endpoints y Métodos

Endpoint	**Método**	**Retorno**	**Descripción**	**Parámetros**
/resources/videos	GET	application /json	Obtener los vídeos de acuerdo con el filtro (título, autor, descripción, año, mes, día).	QueryParam: name
				QueryParam: author
				QueryParam: description
				QueryParam: year
				QueryParam: month

				QueryParam: day
				QueryParam: orderBy
				QueryParam: orderDir
/stream/{videoId}	GET	video/mp4	Reproducir un vídeo dado.	PathParam: videoId
/incrementViews/{videoId}	PUT	application/json	Incrementar el número de visualizaciones de un vídeo dado.	PathParam: videoId
{baseUrl}/upload	POST	application/json	Subir vídeo y su información (que va en el body de la request) a la plataforma.	FormDataParam: title
				FormDataParam: description
				FormDataParam: length
				FormDataParam: authorId
				FormDataParam: uploadedInputStream
				FormDataParam: fileDetail
/details/{videoId}	GET	application/json	Obtener la información del vídeo (título, autor, descripción, año, mes, día).	PathParam: videoId
/update/{videoId}	PUT	application/json	**Extra:** Actualizar el título o descripción de un vídeo ya subido a la plataforma.	PathParam: videoId
				FormParam: title
				FormParam: description

2.4. Funcionalidades

- **Gestión de usuarios:**
  - Inicio de sesión
  - Registro de usuarios
- **Gestión de vídeos:**
  - Reproducir vídeo
  - Subir vídeo
  - Eliminar vídeo

- Obtener vídeo(s) (con filtros)
- Actualizar campos del vídeo
- Incrementar número de visitas del vídeo

2.5. Aclaraciones

Algunos ficheros como el GetVideosServlet.java y VideoServlet.java aún siguen existiendo en el código del proyecto pero ya no se usan. Pertenecen a la entrega anterior, ya sea porque fue un extra que hicimos antes de la entrega 2 o porque en ese momento la arquitectura de la aplicación entera era distinta. Como acordamos, no pasa nada por dejar el código que ya teníamos hecho.

2.6. Extras Realizados

- **Autenticación para utilizar el servicio REST:** hemos añadido un sistema sencillo de autenticación para poder utilizar el servicio REST. Cuando un servlet quiera acceder al servicio REST necesita una apikey. El middleware ApiKeyAuthFilter.java comprueba que la request tenga la apikey correcta.
- **Modificación de los campos de un vídeo ya subido:** hemos hecho que sea posible modificar el título y la descripción de un vídeo que ya hayamos subido. Hemos pensado que poder cambiar el autor, el vídeo en sí o la fecha de creación harán que deje de ser el mismo vídeo, de modo que sólo permitimos la modificación del título y la descripción que el usuario le ha querido poner a ese vídeo.
- **Mejoras visuales:** hemos hecho muchas mejoras visuales a la UI, como por ejemplo hemos añadido un footer y un header (footer.jspf, header.jspf). También hemos cambiado gran parte del CSS para que quede más profesional.

3. Repositorios de código consultados

Nuestros repositorios de código propios de la asignatura de GEI-AD.

4. Bibliografía consultada

https://stackoverflow.com/questions/43157/easy-way-to-write-contents-of-a-java-inputstream-to-an-outputstream

https://www.youtube.com/watch?v=4DY46f-LZ0M&t=1699s

https://stackoverflow.com/questions/51438/how-to-get-a-files-media-type-mime-type

https://www.baeldung.com/exception-handling-for-rest-with-spring

https://docs.oracle.com/javase/tutorial/networking/urls/readingWriting.html

https://www.baeldung.com/convert-input-stream-to-string

https://docs.oracle.com/javase/8/docs/api/java/net/HttpURLConnection.html

https://stackoverflow.com/questions/2793150/how-to-use-java-net-urlconnection-to-fire-and-handle-http-requests