Práctica DAW – Videoclub online

Objetivo

Poner en práctica los conocimientos adquiridos en la asignatura **Diseño de Aplicaciones Web**, en los tres aspectos cubiertos durante la asignatura: *frontend*, *backend* y despliegue.

Enunciado

Se propone implementar una aplicación web cuya función es proporcionar contenidos multimedia a sus usuarios (**videoclub online**). Asimismo, se pretende que la aplicación pueda ser fácilmente desplegada en arquitecturas *cloud* de manera automática, por lo que se requerirá de una serie de *scripts* que automaticen dicho proceso.

Parte 1. Aplicación Web: *frontend* y *backend*

En este primer apartado se cubren los requisitos funcionales de la primera parte de la práctica, relativos al desarrollo del *frontend* y el *backend* de la misma.

En la web habrá dos tipos de roles para los usuarios:

- USER: Para usuarios consumidores de contenidos multimedia
- ADMIN: Para usuarios administradores

Los casos de uso que se deben implementar están representados en el siguiente diagrama UML:

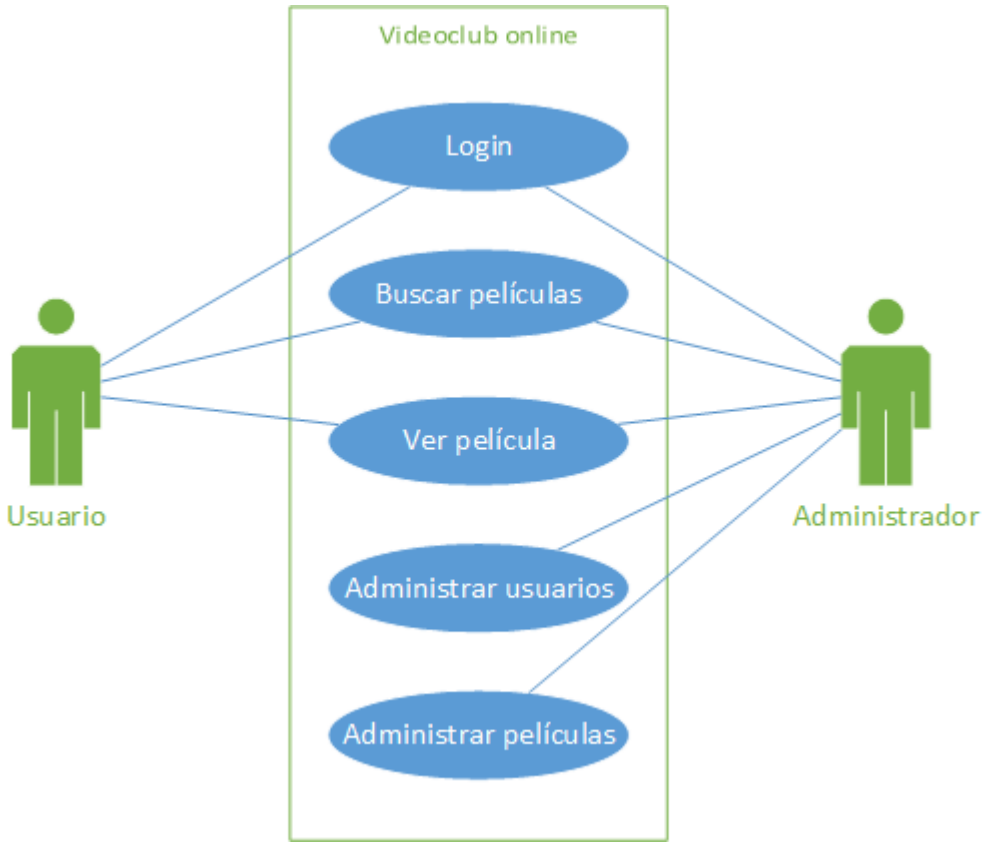

Figura 1 - Diagrama de casos de uso de la aplicación

Login. Todos los usuarios tendrán que estar autenticados en el sistema. Para ello, se usará un formulario que pedirá las credenciales de acceso (nombre de usuario y contraseña). Estas credenciales serán verificadas por el servidor. Todos los usuarios del sistema serán almacenados de forma persistente en una base de datos en el lado servidor.

Buscar películas. Una vez autenticado, todos los usuarios podrán buscar películas. Las películas serán modeladas como entidades persistentes en la base de datos del sistema. La función de búsqueda se realizará al menos usando el campo de **nombre de película** (aunque el alumno podrá añadir tantas opciones de búsqueda como desee).

Cada película deberá almacenar, al menos, la siguiente información:

- Nombre de la película
- URL del contenido de la película (vídeo)
- Descripción
- Año
- Director
- Reparto de actores
- URL de la portada
- Valoración

Ver película. Tras la búsqueda de películas se podrán visualizar las mismas. Para ello se usará el campo "URL del contenido de la película (vídeo)" definido en la entidad película. No es necesario que la película enlace con su contenido real. Para este prototipo de la aplicación, podemos usar contenidos de YouTube similares o relacionados como, por ejemplo, el tráiler de la película en cuestión. Así, si hemos almacenando la película "Interstellar" en nuestro sistema, como contenido podemos almacenar la URL usada para incrustar el vídeo correspondiente de YouTube:

https://www.youtube.com/watch?v=UoSSbmD9vqc

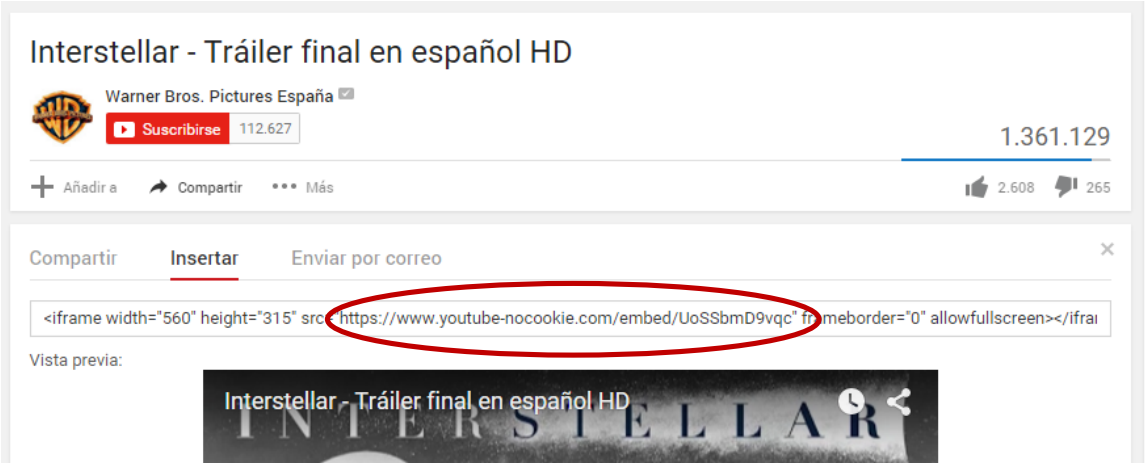

Figura 2 - Ejemplo de contenido de película

Este código URL será almacenado en la base de datos y posteriormente enviado al cliente, que directamente lo añadirá a la página HTML para que el navegador muestre el vídeo. Esto puede hacerse, por ejemplo, con el siguiente código:

```
<iframe width="560" height="315"
        src="https://www.youtube-nocookie.com/embed/UoSSbmD9vqc"
        frameborder="0" allowfullscreen></iframe>
```

Administrar usuarios. Este caso de uso sólo podrá ser llevado a cabo por usuarios de tipo ADMIN. Ten en cuenta que los usuarios normales no tienen la capacidad de darse de alta en el sistema. Esta función (por simplicidad) es realizada por el usuario administrador.

El administrador (usuario de tipo ADMIN) deberá poder realizar operaciones CRUD para los usuarios (usuarios de tipo USER) del sistema. Por lo tanto, el administrador deberá poder dar de alta, de baja, modificar y eliminar usuarios. La información asociada a cada usuario se almacenará en la base de datos del sistema. Esta información deberá ser al, menos:

- Nombre de usuario
- Contraseña (no se debe almacenar en claro en la base de datos)
- Correo electrónico

El usuario administrador debe existir en la base de datos. Para conseguirlo, podemos o bien insertarlo manualmente mediante las herramientas de gestión de la BBDD seleccionada o bien hacerlo programáticamente, por ejemplo, en el arranque de la aplicación (comprobando que el usuario no haya sido insertado previamente).

Administrar películas. Este caso de uso sólo podrá ser llevado a cabo por usuarios de tipo ADMIN. Los administradores deberán poder realizar operaciones CRUD en las películas.

Para dar de alta una película, los administradores rellenarán obligatoriamente los campos "**Nombre de la película**" y "**URL del Contenido de la película**". La idea es que, cuando se vaya a leer una película por otro usuario de la web, si el resto de campos están vacíos se use un servicio REST externo para obtener dicha información, esto es:

- Descripción
- Año
- Director
- Reparto de actores
- URL de la portada
- Valoración

Hay varios servicios REST públicos para consultar información de películas. Por ejemplo:

- Rotten Tomatoes: http://developer.rottentomatoes.com/docs/read/Home
- IMDb REST API: http://imdb.wemakesites.net/documentation.html

Puedes usar una de estas dos APIs o buscar otra alternativa. En cualquier caso, la idea es crear un cliente REST para estos servicios desde el lado servidor que consultarán al servicio REST de nuestra elección para obtener la información anterior (descripción, año, director, reparto, portada, valoración) y almacenarla localmente en nuestra BBDD. Esa información será, posteriormente, enviada del servidor al cliente.

Parte 2. Despliegue de la aplicación

En este segundo apartado se presentan los requisitos funcionales para la segunda parte de la práctica, relativos al despliegue de la aplicación en un entorno *cloud*.

En concreto, esta segunda parte consistirá en preparar una serie de scripts basados en *Vagrant* para desplegar y provisionar la aplicación web descrita anteriormente. Para ello, el alumno tendrá total libertad para escoger la arquitectura final de la aplicación, siempre que se cumplan los siguientes requisitos mínimos:

- La arquitectura deberá contar con al menos dos servidores independientes.
- La instalación de todo el software necesario habrá de hacerse de manera automática.
- La imagen base que se usará para un despliegue local (con varias instancias de *VMs VirtualBox*) será: atorre/daw (esta imagen está disponible públicamente en los servidores Atlas de Hashicorp).

A partir de este punto, el alumno podrá escoger las funcionalidades, de las vistas en clase, que implementará, siendo las anteriormente descritas las mínimas imprescindibles para aprobar la práctica. Se valorará positivamente la complejidad de la arquitectura seleccionada (por ejemplo, mediante la inclusión de un balanceador de carga), el uso de cachés para las consultas, el uso de tecnologías de provisión avanzadas en lugar de *shell scripts*, o el despliegue en un entorno *cloud* real como *Azure* (en cuyo caso la imagen base será, obviamente, una de las disponibles en *Azure*).

Implementación

Los requisitos de implementación de la primera parte de la práctica (*frontend* y *backend*) son los siguientes:

- Se usará Java en el lado servidor, concretamente Spring Boot y el resto tecnologías que hemos visto en clase.
- Se usará Maven para el empaquetado de la aplicación.
- En la parte cliente se deberá usar HTML y Bootstrap. Si fuese necesario usar JavaScript, se usará jQuery.
- La presentación se hará usando Spring MVC con Thymeleaf como tecnología de plantillas.
- La aplicación deberá disponer obligatoriamente de una base de datos. Puedes usar cualquier implementación de base de datos que tenga un *binding* para Spring.
- El cliente REST debe estar implementado en Spring REST Template o Retrofit.

Del mismo modo, los requisitos de implementación de la segunda parte de la práctica (despliegue) son los siguientes:

- El despliegue de las máquinas virtuales en que el alumno haya decidido dividir su aplicación deberá hacerse con *scripts* de *Vagrant* de forma completamente automática.
- La provisión de servicios podrá hacerse con la tecnología que se desee, aunque se valorará positivamente que se use alguna de las arquitecturas avanzadas descritas en clase y no con simples *shell scripts*.

- Si la aplicación se despliega sobre *Azure*, ésta debe ser accesible a través de una IP pública.

Evaluación

La práctica se evaluará sobre un máximo de 10 puntos.

La entrega de la práctica se realizará a través del **Moodle** de la asignatura y constará de un único fichero comprimido que deberá incluir todo el código para poder probar la aplicación y los scripts para desplegarla. Para reducir el tamaño del fichero y evitar problemas con Moodle, tomad la precaución de **no adjuntar las clases Java compiladas**.

La práctica se realizará por **parejas** y deberá entregarse, como último día, el **29 de mayo de 2016**. Será obligatorio realizar una defensa de la práctica, cuya fecha se comunicará posteriormente en función de la disponibilidad de los horarios de evaluación. La nota de la práctica, tal y como figura en la guía de aprendizaje de la asignatura, está compuesta por los siguientes factores:

- 75% nota del profesor al trabajo
- 25% nota de la presentación

Asimismo, la presentación se evaluará de la siguiente manera:

- 1/3 calidad de presentación
- 1/3 autoevaluación
- 1/3 evaluación de pares (nota puesta por los compañeros)

La práctica será evaluada considerando los siguientes criterios:

1. La aplicación tiene que funcionar correctamente siguiendo los requisitos establecidos en el enunciado.
2. Se valorará positivamente que la aplicación sea fácil de usar y que el diseño sea adecuado.
3. Se valorará positivamente un código fuente limpio, que esté estructurado y sea modular.

Defensa de la práctica

Esta defensa consistirá en una presentación oral de unos 10 minutos (incluidas preguntas, aproximadamente dos minutos) en la que se detallarán los resultados de la práctica. Se podrá usar una presentación tipo PowerPoint para guiar el discurso. Ambos componentes del grupo deberán compartir el tiempo de presentación.

Los puntos que debe contener esta presentación son los siguientes:

- Arquitectura de la aplicación (diagrama de componentes Spring de lado servidor).
- Breve descripción de la implementación (entidades persistentes, cliente REST, etc.).
- Arquitectura de servidores en los que se ha desplegado la aplicación y características del mismo.
- Demostración de uso de la aplicación, o en su defecto capturas representativas de la aplicación (se primará la demostración en vivo, dentro de la arquitectura propuesta).

- Dificultades encontradas.
- Conclusiones.