Universidad de los Andes

Ingeniería de Sistemas y Computación

ISIS1206 - Estructuras de Datos

Ejercicio N16 – QueVideo.com

Cupi2

Enunciado

Se quiere diseñar e implementar un sistema Web de gestión de videos para múltiples usuarios. Un usuario debe poder registrarse e ingresar al sitio. El sistema debe permitir buscar videos bajo una o varias palabras clave y los resultados de sus búsquedas deben poder ser guardados. A su vez, un usuario debe poder, dado un video, saber en cuáles índices aparece. Existe también, además de los usuarios normales, un usuario administrador. Este debe ser capaz de consultar ciertas estadísticas sobre el comportamiento del sitio como la cantidad y/o la lista de usuarios registrados, los usuarios activos en un momento determinado y algunas estadísticas particulares sobre los videos con que trabajan los usuarios.

El problema

El objetivo de este ejercicio es analizar, diseñar e implementar el sistema de QueVideo y su interfaz web, utilizando estructuras de datos en memoria principal.

El trabajo sobre las estructuras de datos genéricas continúa y se debe utilizar esta librería para la implementación del mundo del problema.

El sistema debe ser implementado como una aplicación WEB accesible desde cualquier navegador con una interfaz amigable para el usuario.

Asegúrese de leer y entender completamente el enunciado antes de resolver el ejercicio.

Requerimientos funcionales

El programa debe soportar los siguientes requerimientos funcionales:

R1. Permitir al usuario registrarse en la red: Se proporcionan los datos básicos para un usuario y se registra en el sistema

R2. Ingresar al sistema proporcionando un usuario y una contraseña: Debe existir una pantalla de ingreso al sistema que pida el usuario y la contraseña. Si la autenticación se realiza correctamente, debe llevar a la página de inicio, si no, debe indicarle al usuario que hubo un problema. En la página de inicio debe aparecer una opción para visitar lista de contactos.

También debe aparecer una opción para agregar una foto y la lista de todas las fotos usuarios. Finalmente, debe aparecer la opción de buscar un usuario.

R3. Buscar video por palabras clave: Se deben poder buscar videos por el contenido de su nombre. Por ejemplo si la búsqueda se hace por "**Like**", deberían aparecer en pantalla videos cuyo nombre contenga por tal palabra ("**Like** me", "What is **like**", etc.).

R4. Guardar el resultado de una búsqueda: Al realizar una búsqueda, debe poder guardarse el resultado de la misma. Su nombre es el criterio de búsqueda utilizado. El conjunto de todas las búsquedas guardadas conforma un índice de búsquedas.

R5. Buscar sobre el índice: Cuando se realice una búsqueda, debe verificarse primero el índice, antes de recurrir a la búsqueda del requerimiento **R3**.

R6. Dado un video, indicar en cuáles resultados de búsqueda guardados en el índice aparece.

R7. Hacer *like* a un video.

R8. Mostrar al usuario todos los videos a los que ha dado *like*.

R9. Mostrar estadísticas del sitio: Al iniciar como usuario administrador, se deben ver todos los usuarios registrados, todos los que están en línea en un momento determinado, los videos 10 con más *likes* y los 20 videos más vistos.

R10. Comentar un video: Se deben poder adicionar un comentario sobre un video en particular.

Restricciones de diseño

1. El diseño se debe hacer **MINIMIZANDO el tiempo de ejecución de todas las operaciones y el espacio que ocupan las estructuras de datos, garantizando que el programa pueda evolucionar**.

2. En particular, se deben justificar las decisiones de diseño tomadas, se debe explicar claramente en un **documento de diseño** por qué se escogieron las estructuras de datos utilizadas, teniendo en cuenta los requerimientos descritos anteriormente. Adicionalmente, es importante que en este documento (y definido desde antes de la implementación) esté el mapa del sitio, en este se debe indicar cuáles páginas existen, desde qué página se posible navegar a cuáles y cuáles son las funcionalidades proporcionadas en cada una de ellas.

3. El sistema debe ser **persistente**. Recuerde que el objetivo del ejercicio es resolver los requerimientos con las estructuras en memoria principal, por este motivo **no se acepta el uso de bases de datos**. Debe utilizar XML para la persistencia de la información.

4. El sistema debe ser desarrollado como una aplicación WEB. Para ello, debe definir su modelo del mundo como un Singleton, un conjunto de páginas estáticas y un conjunto de páginas dinámicas siguiendo una arquitectura MVC.

5. La interfaz gráfica **debe ser implementada utilizando Servlets** y ser usable y amigable. Se debe tener una navegación clara (siempre poder volver a la página de inicio) y utilizar correctamente el diseño, la tipografía y los colores para presentar los diferentes contenidos.

6. El diseño debe estar desacoplado con interfaces.

7. Debe implementar las estructuras de datos **genéricas** en un **paquete aparte**. Las estructuras de datos se deben probar con **pruebas unitarias**.

8. Debe utilizar árboles 2_3 y/o Tries.

9. El sistema debe utilizar ant para su compilación y ejecución

10. No se puede utilizar Java Collections ni Cupi2 Collections.

11. Debe utilizar el API de Vimeo para la gestión de los videos. Puede consultar el **documento adjunto** como guía para su uso.

Entregables

Entrega 1. – Lanzamiento (25%) (Fecha: 23 Oct)
E1. Documento de Análisis y Diseño del problema (Incluye Complejidad Temporal)
E2. Documento de lanzamiento (Definición, priorización y planeación de tareas).
E3. Mapa de navegación de la aplicación (mostrando cómo se exponen al usuario los diferentes requerimientos, cuáles páginas son dinámicas y estáticas).
E4. Prueba(s) de Autenticación al servidor vimeo.com (Obtención y uso de llave de autenticación a servidor vimeo.com)
E5. Prueba(s) de Consulta de un Video al servidor vimeo.com y mostrar el contenido de respuesta recibida (contenido XML o JSON). Esta prueba No requiere desarrollo de aplicación Web.

Entrega 2. – Intermedia (30%) (Fecha: 30 Oct)
E1. Documento de Análisis y Diseño del problema (Incluye Complejidad Temporal) – versión avance
E2. Estructuras de datos a utilizar implementadas en un proyecto aparte usando genericidad. Uso de *ant* para la generación de la librería de Estructuras de Datos.
E3. Pruebas unitarias de las nuevas estructuras de datos.
E4. Implementación de Interfaz Web para los requerimientos R1, R2 y R3. La página Web de inicio corresponde a una página de registro de usuarios y/o autenticación de usuarios.
E5. Implementación completa de los requerimientos R1, R2 y R3. Se califican únicamente aquellos requerimientos que sean funcionales desde la interfaz gráfica. Los requerimientos deben funcionar accediéndolos desde un navegador Web y ejecutándose en una aplicación web. Uso de *ant* para la compilación y despliegue de la aplicación Web.

Entrega 3. – Final (45%)(Fecha: 6 Nov)

E1. Documento de Análisis y Diseño del problema (Incluye Complejidad Temporal) – versión final

E2. Documento de lanzamiento (Definición, priorización y planeación de tareas) – versión completa

E3. Mapa de navegación de la aplicación (mostrando cómo se exponen al usuario los diferentes requerimientos, cuáles páginas son dinámicas y estáticas) – versión final.

E4. Implementación completa de la Interfaz Web. La página Web de inicio corresponde a una página de registro de usuarios y autenticación de usuarios.

E5. Implementación completa de los requerimientos funcionales. Se califican únicamente aquellos requerimientos que sean funcionales desde la interfaz gráfica. Los requerimientos deben funcionar accediéndolos desde un navegador Web y ejecutándose en una aplicación web. Uso de *ant* para la compilación y despliegue de la aplicación Web.

E6. Javadoc de la aplicación.

El mejor ejercicio de cada sección tendrá un bono especial sobre la nota del nivel.