Proyecto #2 - Odyssey

Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Algoritmos y Estructuras de Datos 1
I Semestre 2018
Valor 20%

Objetivo general

- ➔ Implementar una aplicación cliente-servidor para gestionar una biblioteca musical y reproduzca música mediante streaming.

Objetivos específicos

- ➔ Aplicar conocimientos de búsquedas
- ➔ Aplicar conocimientos sobre árboles
- ➔ Implementar y diseñar una solución cliente-servidor
- ➔ Aplicar OO en C# .NET y Java

Descripción del problema

El proyecto consiste en la implementación de una aplicación cliente-servidor. El Cliente se denomina Odyssey y el servidor Odyssey Server. La comunicación entre el cliente y servidor se realiza mediante socket utilizando XML como formato de mensajes.

El cliente

Es una aplicación de escritorio escrita en C# .NET que permite gestionar una biblioteca de archivos musicales (MP3). Posee las siguientes funciones:

1. **Gestionar la biblioteca musical**: El cliente muestra un diálogo de selección de carpetas/archivos que el usuario puede agregar a la biblioteca. Cuando los archivos se agregan, se cargan al servidor que se encarga de almacenarlos.
    a. El usuario puede eliminar canciones de la biblioteca, eliminandolos también del servidor.
    b. El usuario puede modificar la metadata de cada canción (género, artista, álbum, año, letra y demás relevantes) y sincronizarlo con algún proveedor de metadata externo (Last.FM, Spotify, entre otros)
    c. El usuario puede calificar la canción con estrellas.
    d. El usuario puede ordenar la biblioteca musical por:
        i. Nombre de la canción utilizando QuickSort
        ii. Nombre del artista utilizando RadixSort
        iii. Nombre del álbum utilizando BubbleSort

    El ordenamiento se realiza en el servidor, es decir, el cliente envía una petición al servidor para obtener la lista de canciones ordenada por el campo indicado

El cliente requiere que el usuario se registre antes de usar la aplicación. El cliente presenta una ventana de login/registro. El registro se valida y guarda en el servidor.

Dado que la biblioteca musica puede crecer mucho y no se quiere tener que esperar mucho por la lista de canciones, el servidor pagina la lista de canciones al enviarla al cliente. El cliente tiene un mecanismo de páginas o lazy loading.

2. **Reproducir música**: El usuario puede reproducir música desde el servidor mediante streaming. El servidor no envía el archivo de audio completo, sino que tiene capacidades para hacer streaming real. El usuario puede adelantar o atrasar la canción, lo que será considerado por el servidor para enviar los *chunks* del archivo correspondiente.

   El cliente muestra un ecualizador gráfico (animación acorde al ritmo) opcionalmente implementado por los estudiantes que reaccione según la música que se reproduce. De ser implementado por los estudiantes, se darán puntos extra a criterio del profesor.

3. **Gestionar amigos y recomendaciones**: El usuario puede agregar amigos que ya existan en el servidor y puede recomendarle canciones. Las recomendaciones deben aparecer como mensajes entrantes en el cliente.

4. **Buscar canciones**: el cliente permite que el usuario pueda buscar canciones por distintos campos:
   a. Nombre de la canción
   b. Nombre del artista
   c. Nombre del album
   d. Letra

   La consulta se envía al servidor y este retorna los resultados. Ver más sobre esto adelante.

El servidor

Es una aplicación escrita en Java que escucha conexiones entrantes por socket. Las funcionalidades principales son:

1. **Manejo de usuarios**: El servidor permite que la aplicación cliente autentique y registre usuarios. A la hora de registrar usuarios se requieren los siguientes datos:
   a. Nombre de usuario. Identifica a los usuarios y son únicos
   b. Nombre y Apellidos.
   c. Edad.
   d. Géneros musicales favoritos
   e. Contraseña.
   f. Amigos

Los usuarios se almacenan en un archivo JSON con todos los datos definidos. Cuando el servidor inicia, se carga este archivo en memoria en un árbol binario de búsqueda. Cuando se agrega un usuario, se actualiza en el archivo y en el árbol binario. La contraseña se guarda encriptada mediante la función de hash MD5.

2. **Gestión de la biblioteca musical**: El servidor permite gestionar la biblioteca musical:
    a. Recibe archivos de audio enviados a través de socket y los guarda en una carpeta predefinida. Además de guardar los archivos en dicha carpeta, se almacena la metadata de las canciones en un archivo JSON. Cuando el servidor inicia, crea los siguientes índices:
    b. Realiza streaming de los archivos de audio
    c. Permite actualizar la metadata de los archivos de audio según como se le envíe desde el cliente. También puede sincronizar la metadata con el proveedor externo cuando se le indique por el cliente.
    d. Permite eliminar archivos de audio

3. **Gestión de amigos y recomendaciones**: el servidor administra la lista de amigos de cada usuario y se los muestra en el cliente. El servidor también tiene un mecanismo para enviar las recomendaciones musicales al cliente.

4. **Consultas**: El servidor recibe consultas de búsquedas de canciones del cliente. Cuando el servidor inicia, crea los siguientes índices para la biblioteca musical:
    a. Nombre de la canción: Utiliza un árbol B
    b. Nombre del artista: Utiliza un árbol AVL
    c. Nombre del álbum: Utiliza un árbol Splay
    d. Letra: Indexado por algún mecanismo diseñado por los estudiantes

Documentación requerida

1. Se deberá documentar el código fuente utilizando JavaDoc.
2. Se deberá entregar un documento que contenga:
    a. Todas las partes estándar: Portada, índice, introducción, conclusión, bibliografía y anexos.
    b. El cuerpo del documento debe contener:
        ➔ Breve descripción del problema
        ➔ **Planificación y administración del proyecto**
            ◆ Lista de features e historias de usuario identificados de la especificación.
            ◆ Distribución de historias de usuario por criticalidad y secuencia de uso.
            ◆ Minimal system span.
            ◆ Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un desarrollo incremental y especificar el responsable de cada una.

- ➔ **Diseño**
  - ◆ Diagrama de componentes.
  - ◆ Diagrama de arquitectura.
  - ◆ Diagrama de secuencia (Seleccionar al menos 3 Historias de Usuario).
  - ◆ Diagrama de clases inicial.
  - ◆ Diagrama de clases final (generado utilizando el IDE).
- ➔ **Implementación**
  - ◆ Descripción de las estructuras de datos desarrolladas.
  - ◆ Descripción detallada de los algoritmos desarrollados.
  - ◆ Problemas encontrados: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo. Incluye descripción detallada, intentos de solución sin éxito, solución encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.

Aspectos operativos y evaluación:

1. **Fecha de entrega: De acuerdo al cronograma del curso.**
2. El proyecto tiene un valor de 20% de la nota del curso.
3. El trabajo es **en grupos de 3 personas.**
4. Es obligatorio utilizar un manejador de versiones del código. Puede ser git o svn. Se revisará que cada commit lleve comentarios relevantes relacionados con alguna tarea identificada en la sección de planificación.
5. **Los proyectos que no estén debidamente integrados, no serán revisados**.
6. El código tendrá un valor total de 65%, la documentación 30% y la defensa 5%.
7. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
8. Se evaluará que la documentación sea coherente, acorde a la dificultad/tamaño del proyecto y el trabajo realizado, se recomienda que realicen la documentación conforme se implementa el código.
9. La nota del código NO podrá exceder en 35 puntos la nota de la documentación, por lo cual se recomienda documentar conforme se programa.
10. Se debe enviar el código (preliminar) y la documentación a más tardar a las 23:59 del día de la fecha de entrega al email del profesor. **Se debe seguir el formato del Subject descrito en el Programa del Curso.**
11. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
12. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la cita revisión oficial, momento en el cual deben enviar un email con el código fuente, si no lo hacen se asumirá que la versión oficial del código fue la enviada el día de la fecha de entrega junto con la documentación. **Se debe enviar en el mismo correo del punto 8.**

13. Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones:
    a. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
    b. Si no se utiliza un manejador de código se obtiene una nota de 0.
    c. Si el código y la documentación no se entregan en la fecha indicada se obtiene una nota de 0.
    d. Sí el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
    e. El código debe ser desarrollado en Java, en caso contrario se obtendrá una nota de 0.
    f. Si no se siguen las reglas del formato de email se obtendrá una nota de 0.
    g. La nota de la documentación debe ser acorde a la completitud del proyecto.
14. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto. El único requerimiento que se consultará durante la defensa del proyecto es el diagrama de clases, documentación interna y la documentación en el manejador de código.
15. Cada estudiante tendrá como máximo 15 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
16. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
17. Cada estudiante es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberán avisar al menos 2 días antes de la revisión a el profesor para coordinar el préstamo de estos.
18. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.