Trabajo final – Parte 2

La segunda parte del trabajo transforma la aplicación de gestión de biblioteca de música desarrollada en la Parte 1 en un sistema cliente-servidor basado en el modelo de sincronización por usuario.

El objetivo principal de esta etapa es implementar las comunicaciones de red y la concurrencia, permitiendo que múltiples usuarios distintos puedan gestionar sus bibliotecas de música de forma centralizada y segura, utilizando el servidor como una copia de seguridad y punto de control.

Recordatorio: todas las entregas del trabajo se realizarán por parejas. Sin embargo, la nota final de prácticas será de carácter individual, ponderando la nota del trabajo por la calificación obtenida en el examen sobre el mismo, tal y como se indica en la Presentación de la asignatura.

Descripción del funcionamiento general

En esta parte, el proyecto constará de dos programas principales:

1. Servidor (*servidor.py*): almacena de forma persistente la metadata y los archivos MP3 de cada usuario. Gestiona la conexión concurrente de múltiples usuarios y garantiza que cada cuenta esté bloqueada mientras está en uso.
2. Cliente (*cliente.py*): proporciona la interfaz de consola (como el antiguo *app.py*), y ejecuta las operaciones de sincronización al inicio y al final de la sesión.

La aplicación operará bajo un modelo de "cerradura digital", garantizando que los datos de un usuario nunca sean modificados simultáneamente desde dos ubicaciones. El flujo de una sesión de usuario es el siguiente:

1. **Conexión y Bloqueo:** El cliente se conecta y envía su nombre de usuario. El servidor verifica que la cuenta no esté activa. Si la cuenta está libre, el servidor la **bloquea** y acepta la conexión.
2. **Sincronización de Descarga (Inicio de Sesión):** Una vez conectado, el cliente solicita y **descarga** su biblioteca completa (metadata y todos los archivos MP3 asociados) desde el servidor. Con estos datos, la aplicación cliente carga su instancia local de *PlataformaMusical*.
3. **Trabajo Local:** El usuario interactúa con la aplicación por consola (menús de gestión de canciones, listas, reproducción). Todas las modificaciones se realizan **únicamente en la copia local** de la biblioteca.
4. **Sincronización de Subida (Cierre de Sesión):** Al seleccionar la opción **Salir**, el cliente:
    a. **Serializa y sube** la metadata final de su biblioteca al servidor.
    b. **Sube los archivos MP3** asociados a su biblioteca.
    c. Finalmente, el cliente notifica al servidor para que **desbloquee la cuenta** de usuario, permitiendo futuras conexiones desde cualquier PC.

Requisitos de la entrega

Requisitos del servidor

El servidor debe ser capaz de atender a múltiples usuarios distintos de forma concurrente (mediante hilos) y gestionar el almacenamiento de datos.

Concurrencia y Bloqueo de Usuario

- El servidor debe escuchar nuevas conexiones y **lanzar un hilo por cada cliente**.
- **Autenticación sencilla**: El cliente debe enviar un nombre de usuario al conectarse.
- Bloqueo: El servidor debe garantizar que **solo un cliente a la vez puede estar conectado con una cuenta de usuario específica**. Si un segundo cliente intenta conectarse con la misma cuenta, debe ser rechazado inmediatamente. (El servidor debe mantener un registro de usuarios activos/bloqueados).

Gestión de Almacenamiento y Archivos MP3

- El servidor debe tener una estructura de carpetas (ej. datos_server/) para **almacenar la biblioteca de cada usuario** (metadata y archivos MP3).
- **Metadata**: los metadatos de la biblioteca (información de canciones y listas) debe serializarse y **almacenarse en el servidor utilizando un archivo JSON por usuario**. Para la lectura y escritura de archivos JSON, se debe utilizar la librería *json* (*import json*)
- **Archivos MP3**: El servidor debe almacenar todos los archivos MP3 subidos por **cada usuario en carpetas separadas** (ej. datos_server/usuario_X/).

Requisitos del cliente

El cliente mantendrá la interfaz de consola de la Parte 1. Por tanto, toda la lógica de la aplicación se ejecutará de forma local.

Flujo de Sincronización y Sesión

El ciclo de vida del cliente debe seguir este flujo:

1. **Inicio y Conexión:** El cliente pide el nombre de usuario y se conecta al servidor, enviando su credencial.
2. **Sincronización Inicial (Descarga):**
    a. Si la conexión es aceptada (usuario desbloqueado), el cliente debe recibir la metadata de su biblioteca desde el servidor.
    b. Descarga de archivos MP3**:** El cliente también debe descargar todos los archivos MP3 asociados a su biblioteca para poder realizar la reproducción local. A partir de los metadatos recibidos, el cliente irá solicitando los archivos uno a uno.
    c. La aplicación crea o carga la instancia local de *PlataformaMusical* con estos datos.
3. **Trabajo local:** Una vez sincronizado, la aplicación funciona exactamente igual que en la Parte 1 (la lógica se ejecuta en el cliente).
4. **Cierre y Sincronización Final (Subida):**
    a. Al seleccionar la opción *"0) Salir"*, el cliente serializa el estado final de su *PlataformaMusical*.
    b. Subida de Metadata y archivos MP3**:** El cliente debe subir al servidor:

   i. La metadata actualizada.
   ii. Los archivos MP3 de las canciones de su biblioteca.
c. Una vez completada la subida, el cliente cierra la conexión y notifica al servidor para que desbloquee la cuenta de usuario.

Transferencia de Archivos

El cliente debe implementar la lógica para subir y descargar archivos MP3 desde y hacia el servidor. (utilizando el protocolo de "enviar tamaño, luego enviar datos en *chunks*").

Para garantizar una transferencia de archivos binarios (MP3) confiable entre el cliente y el servidor, es obligatorio utilizar un protocolo simple de dos pasos sobre el socket TCP.

Este protocolo es necesario porque el socket opera como un flujo de bytes continuo (stream) y no tiene un mecanismo inherente para reconocer el final de un archivo.

El proceso de transferencia debe seguir siempre esta secuencia:

1. Paso de Metadatos de Archivo (Tamaño y Nombre): El emisor (ya sea el cliente subiendo o el servidor descargando) debe enviar primero los metadatos necesarios para que el receptor se prepare:
   - **Tamaño del Archivo:** Se debe enviar el tamaño total del archivo en *bytes* (un número entero, int).
   - **Nombre del Archivo:** El nombre del archivo se envía a continuación (como una cadena de texto, str) para que el receptor sepa con qué nombre guardar el archivo localmente.

   El receptor lee estos metadatos para determinar la cantidad exacta de datos que debe esperar en el siguiente paso.

2. Paso de Datos Binarios (contenido del MP3): una vez que el receptor conoce el tamaño total, el emisor procede a enviar el archivo en pequeños bloques o fragmentos (*chunks*) de tamaño fijo:
   a. **Envío por Fragmentos:** El archivo binario se lee en el lado del emisor usando el modo binario ('rb') y se envía en un bucle utilizando un tamaño de bloque razonable (por ejemplo, 1024).
   b. **Recepción Acumulativa:** El receptor entra en un bucle para recibir datos, escribiéndolos en un nuevo archivo abierto en modo binario ('wb'). El receptor debe llevar un conteo de los *bytes* recibidos (bytes_received).
   c. **Finalización de la Transferencia:** El receptor debe salir del bucle de recepción solo cuando bytes_received sea **exactamente igual** al tamaño total del archivo.

<u>Consideraciones clave</u>

- **Archivos Binarios:** Tanto en el cliente como en el servidor, los archivos MP3 deben abrirse en modo **binario** ('rb' para lectura y 'wb' para escritura).
- **Recepción del último fragmento de un MP3**: el último bloque de cada archivo puede no tener la misma longitud que el tamaño de bloque fijado. Hay que tenerlo en cuenta en el receptor, para **leer únicamente los bytes restantes**.

Ejecución de la aplicación

Para iniciar la aplicación, en primer lugar, se deberá ejecutar el servidor: *python3 servidor.py <puerto>*

Posteriormente, cada cliente que quiera conectarse se lanzará de la siguiente forma: *python3 cliente.py <ip_servidor> <puerto_servidor>*

Extensiones

Además del cumplimiento de los requisitos descritos con anterioridad, se valorará positivamente:

- Sincronización final inteligente: En la sincronización final, en lugar de subir toda la metadata y los MP3 de las canciones en la biblioteca, enviar solo los cambios realizados (canciones añadidas/eliminadas, listas modificadas).
- Servidor bien mantenido: cuando un cliente elimine canciones de su biblioteca, asegurarse de que los archivos MP3 correspondientes son eliminados de los datos del servidor.

Entrega del trabajo

El proyecto se entregará a través de Aula Virtual en una tarea habilitada para tal efecto antes del 2 de diciembre a las 23:59. La entrega la efectuarán todos los miembros del grupo, y consistirá en un único archivo .zip que incluya la carpeta entera con todos los archivos del proyecto (*.py*) cuyo nombre seguirá el siguiente formato: parte2_NOMBRE1_APELLIDO1_NOMBRE2_APELLIDO2.zip.