Trabajo final – Parte 3

Esta tercera y última parte del proyecto se centra en la gestión de estados y el versionado de datos mediante la implementación formal de los Tipos Abstractos de Datos (TADs), específicamente la Lista Enlazada y la Pila. Se incorporará una funcionalidad de Deshacer/Rehacer en el cliente y un sistema de copias de seguridad en el servidor.

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.

Requisitos de la entrega

Implementación de Tipos Abstractos de Datos (TADs)

Se requiere la implementación desde cero de las clases que representan los TADs mencionados en los apartados siguientes. Estas clases no deben usar las estructuras de datos nativas de Python (como *list*) en su implementación funcional.

Concretamente, se requieren una Lista Doblemente Enlazada y una Pila. En ambos casos, sólo es obligatorio implementar los métodos necesarios para el funcionamiento del sistema.

Requisitos del servidor

El servidor debe usar el TAD Pila para almacenar las copias de seguridad de la metadata de la biblioteca de cada usuario.

- **Pila de Versiones:** Por cada usuario, el servidor debe mantener una **Pila de Versiones** (utilizando el **TAD Pila**).
- **Generación de versión:** Cada vez que un usuario completa su sesión y sincroniza (subida final), la metadata anterior de la biblioteca se guarda en la pila, y la nueva metadata se convierte en la versión "actual".
- **Estructura de la versión:** Cada elemento apilado debe contener la ruta del archivo JSON de metadata correspondiente, cuyo nombre incluirá una marca de tiempo que lo identifique con el formato *AÑO_MES_DIA_HORAS_MINUTOS_SEGUNDOS*:
    - Ejemplo: *datos_server/usuario_X/biblioteca_2025_12_16_23_59_59.json*

Es imprescindible garantizar que, si el servidor se para y vuelve a arrancar, los clientes que se conecten puedan sincronizarse con la versión más reciente de su biblioteca, preservando además todo su historial.

Requisitos del cliente

El cliente debe utilizar una instancia del TAD Lista Enlazada para gestionar el historial de acciones y permitir al usuario revertir y restaurar las últimas modificaciones realizadas en su biblioteca local.

- **Historial de Estado:** Se utilizará la **Lista Enlazada** para almacenar una secuencia de estados de la biblioteca (*PlataformaMusical*). El dato de cada nodo de la lista representará una versión de la biblioteca antes de una modificación.
- **Puntero de Estado:** Se debe mantener un puntero lógico que indique el **estado actual** que el usuario está viendo/modificando dentro de la Lista Enlazada.
- **Comportamiento:** Cualquier nueva acción de modificación debe **eliminar todos los nodos futuros** (acciones "rehacibles") del puntero de estado hacia el final de la lista, insertar el nuevo estado al final, y mover el puntero al nuevo estado.

Para permitir al usuario Deshacer/Rehacer acciones, el menú principal debe incluir dos nuevas opciones:

- **"Deshacer Última Acción"** (Mueve el puntero de estado una posición hacia atrás en la lista enlazada y carga ese estado).
- **"Rehacer Última Acción Deshecha"** (Mueve el puntero de estado una posición hacia adelante en la lista enlazada y carga ese estado).

Estas opciones se mostrarán al usuario cuando haya acciones que se puedan deshacer/rehacer. De esta forma, si el usuario acaba de conectarse y únicamente ha añadido una canción, podría *deshacer* (cancelar) esa operación, pero no podría *rehacer* nada. En esta situación el menú se verá así:

```
=== Plataforma Musical ===
1) Gestionar canciones
2) Gestionar listas
3) Reproducción
4) Deshacer última acción
5) Rehacer última acción deshecha (no disponible)
0) Salir
```

De forma análoga, la operación número 4 se marcará como *no disponible* cuando proceda.

Ejecución de la aplicación

La ejecución del sistema será idéntica a la requerida en la segunda entrega del proyecto:

En primer lugar, se deberá ejecutar el servidor: *python3 servidor.py <puerto>*

Posteriormente, cada cliente se conecta vía: *python3 cliente.py <ip_servidor> <puerto_servidor>*

Entrega del trabajo

El proyecto se entregará a través de Aula Virtual en una tarea habilitada para tal efecto antes del 16 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: parte3_NOMBRE1_APELLIDO1_NOMBRE2_APELLIDO2.zip.