UTN – FRMDP Mar del Plata ***TUP - Laboratorio 1*** ***Trabajo Práctico Final:*** ***LabTwo Prime Video*** *Octubre 2021*	***Integrantes del grupo***	***Nota***

Introducción

Con el propósito principal de integrar todo lo aprendido en la materia hemos planteado la siguiente problemática:

- **CODIFICAR UN SISTEMA DE USUARIOS Y PELÍCULAS PARA UN POSIBLE NUEVO EMPRENDIMIENTO… LABTWO PRIME VIDEO (CUALQUIER PARECIDO CON AMAZON PRIME VIDEO… ES PURA COINCIDENCIA!!…).**
- Codificar un sistema de Logueo que gestione el sistema desde el punto de vista del **ADMINISTRADOR** del mismo, como desde el punto de vista de los **USUARIOS**.
- Administrar un archivo de Películas y otro de usuarios (Alta, Baja, Modificación, Consulta y Listados)
- Generar y administrar las Playlist de los Usuarios.

Fundamentación

El valor pedagógico de la propuesta se apoya en el aprendizaje colaborativo (se formarán grupos de 3 alumnos) a partir del desarrollo de un proyecto de software. Para que este tipo de proyectos sea más exitoso, deben llevarse a cabo desde un enfoque que facilite alcanzar los objetivos de aprendizaje propuestos.
Una de las ideas centrales es desarrollar competencias profesionales y preparar al futuro programador para el mundo laboral y el trabajo en equipo.
En un ambiente de aprendizaje colaborativo, los estudiantes:

- Construyen conocimiento y en lugar de recibirlos en forma pasiva;
- Se involucran y se comprometen directamente con el descubrimiento de nuevo conocimiento;
- Se exponen a puntos de vista alternativos e ideas contrapuestas, de forma tal que pueden sacar sus propias conclusiones y así transformar conocimientos y experiencias previas y de esta manera comprender con mayor profundidad;
- Transfieren conocimientos y habilidades a nuevas situaciones o circunstancias;
- Se responsabilizan y apropian tanto de su aprendizaje continuo de contenidos curriculares, como del desarrollo propio de competencias;
- Los estudiantes colaboran para el aprendizaje del grupo y el grupo colabora en el aprendizaje individual de estos.

Objetivos

De aprendizaje:

- Gestión de archivos binarios.
- Recursividad.
- Listas enlazadas, simples o dobles
- Arboles binarios.
- Estructura de datos compuestos. (arreglo de listas, listas de listas, listas de árboles, etc.)
- Trabajar en forma colaborativa

Metodológicos:

- Ser capaces de trabajar en un proyecto complejo, aplicando técnicas de desarrollo de software.
- Lograr integrar contenidos de otras asignaturas.
- El grupo deberá ir mostrando el avance sobre el trabajo en clase.

Modo de Evaluación del Trabajo Práctico

- Se establece el desarrollo de un trabajo práctico final, brindando una fecha límite de entrega del mismo.
- Si el sistema está codificado en su totalidad y funciona correctamente, se considerará aprobado con una nota mínima de 6.
- Si el sistema no está codificado en su totalidad (como mínimo un 60 % en cada inciso), se considerará desaprobado y el grupo presentará la versión final en la fecha de recuperatorio.
- En la fecha de recuperatorio deberá cumplir las pautas mínimas establecidas precedentemente para la aprobación de la instancia recuperatoria. Vale aclarar que no puede aprobar de manera directa.
- La aprobación del trabajo práctico estará sujeta a los puntajes considerados en la tabla debajo.
- Es obligatorio la presentación de este trabajo para aprobar la materia.

Al realizar la entrega final, deberán tener en cuenta los siguientes puntos:

- Carpeta completa según lo requerido por la cátedra, que será compartida vía Google Drive con el equipo docente.
- Código del sistema completo, compilado y sin errores
- Explicación presencial del sistema (por clase virtual)

Tabla de puntuación:

Obtenido	10	20	30	40	50	60	70	80	90	100
Nota	1	2	3	4	5	6	7	8	9	10
	Desaprobado					Aprobado		Ap. Directa		

PAUTAS GENERALES

Utilizando el sistema desarrollado en Laboratorio 1, se nos pide realizar una adaptación del mismo para lograr un manejo dinámico de la información, aplicando todos los conceptos trabajados a lo largo del año.

Como metodología de trabajo, se requiere crear un documento de texto en Google Drive que será compartido a todos los miembros del grupo (y también al equipo docente, publicando el link vía campus virtual en el foro correspondiente), con el fin de plasmar los avances del proyecto de forma de construir la siguiente documentación a entregar:

Informe final - Documentación a entregar: [10 puntos]

- Diario de trabajo: Día a día qué actividades se desarrollaron y el responsable de cada una.
- Matriz de soluciones: Que problema tuvieron y cómo lo resolvieron.
- Manual de usuario: Breve explicación de cómo funciona el sistema, pueden usar imágenes, videos, presentaciones, etc.
- Diagrama de estructuras: Esquema de las estructuras utilizadas y sus relaciones.

Asimismo, deberán crear un tablero en www.trello.com para distribuir las tareas entre los integrantes del grupo y trabajar de forma organizada. A medida que avancen con el desarrollo del trabajo, realizarán capturas de pantalla y las adjuntarán al Diario de trabajo. Deberán compartir el tablero con el equipo docente, publicando el link vía campus virtual en el foro correspondiente.

Detalle de estructuras y funcionalidad básicas:

Integración y/o adaptación de funciones del TP Final de Laboratorio 1: [10 puntos]

Importar y adaptar las funciones de Alta, Baja, Modificación, Consulta y Listados de Usuarios y Películas que ya fueron desarrollados en el TP Final de Laboratorio 1.

stUsuario	stPelicula
int id;	int id;
char apellidoYnombres [50];	char nombre [50];
char mail [50];	int año;
int celular;	char genero [30];
char contraseña [8];	char actores [3][50];
Playlist: queda a criterio de cada grupo cómo manejar la persistencia de la lista de películas del usuario.	int calificacion;
int estado;	int estado;

STRUCT USUARIO:

El id es el número UNICO identificatorio de cada usuario. A tal fin, se asigna en forma autoincremental.
Playlist: queda a criterio de cada grupo cómo manejar la persistencia de la lista de películas del usuario.
Estado es un campo booleano en el cual se guarda 1 si el usuario está activo y 0 si el usuario fue dado de baja (concepto de “baja lógica” ya aprendido en Laboratorio 1). Se inicializa en 1, y se pasa a 0 si el Usuario es dado de baja.

STRUCT ADMIN:

Similar a la de un usuario común, con la salvedad de que no manejará una playlist.
Deberá contemplarse algún criterio a elección del grupo que sirva para evaluar que es un administrador.

STRUCT PELICULA:
El id es el número UNICO identificatorio de cada película. A tal fin, se asigna en forma autoincremental.
El género puede ser un string, y para facilitar luego la tarea de comparación para la búsqueda de géneros, puede ser cargado exactamente con los nombres (ej: ACCION, COMEDIA, DRAMA, ROMANTICA, SUSPENSO, TERROR), o puede manejarse con criterios más flexibles a elegir por cada grupo.
La calificación es un número entero del 1 al 5 que representa las estrellas que posee cada película de acuerdo a las opiniones que ha recibido.
Uno de los campos de la estructura película es una matríz de 3x50, un arreglo de palabras de 3 filas, en la cual se guardarán los actores que trabajan en cada películas.
Estado es un campo booleano en el cual se guarda 1 si la Película está actualmente en la lista de Películas del Usuario y 0 si fue dada de baja (concepto de “baja lógica” ya aprendido en Laboratorio 1).

Lista de Películas (lista simple) [15 puntos]

```
typedef struct
{
    stPelicula peli;
    struct nodoListaPelis * sig;
} nodoListaPelis;
```

Deberán codificar todas las funciones necesarias para administrar el TDA Lista Simple, a saber (como mínimo):

inicLista()
crearNodoLista()
agregarAlPrincipio()
agregarAlFinal()
agregarEnOrdenPorNombrePelicula()
mostrarLista() // modularizar
borrarNodoPorIdPelicula()

Árbol de Películas [20 puntos]

```
typedef struct
{
    stPelicula peli;
    struct nodoArbolPelis * izq;
    struct nodoArbolPelis * der;
} nodoArbolPelis;
```

Deberán codificar todas las funciones necesarias para administrar el TDA Árbol, a saber (como mínimo):

inicArbol ()

crearNodoArbol ()
insertarNodoArbol () // ordenado por id película
mostrarArbol () // los 3 recorridos: inOrder, postOrder, preOrder - modularizar
borrarUnNodoArbolPorId ()
buscarPeliculaPorId ()
buscarPeliculaPorNombre ()

cargarArbolDesdeArchivo(): Al iniciar el sistema se deberán cargar todas las películas del archivo sobre un árbol binario ordenado por id, de forma tal que las búsquedas se realicen de forma más eficiente.
Tenga en cuenta que, seguramente, su archivo de películas está ordenado de forma creciente por id, y que si realiza un recorrido secuencial del archivo, la inserción en el árbol no se realizará de una forma óptima. Desarrolle una función (o varias) que logren realizar la inserción en el árbol, logrando que este quede lo más balanceado posible.

Lista de Usuarios (Lista de listas o LDL) [35 puntos]

```
typedef struct
{
    stUsuario user;
    nodoListaPelicula * playlist;
    struct nodoListaUsers * sig;
} nodoListaUsers;
```

Esta lista se cargará de forma automática al iniciar el sistema, con todos los usuarios activos, y a su vez cada uno de ellos con las películas de sus respectivas playlist.
El trabajo en el sistema se realizará sobre esta estructura, y al momento de realizar cualquier modificación (alta de nuevas películas en la playlist, modificación del perfil del usuario, etc), se realizarán las acciones necesarias para actualizar la información sobre la LDL. Luego, una vez que el usuario quiera desloguearse o antes de cerrar el programa, se persistirán los datos en los archivos.

Deberán codificarse todas las funciones necesarias para administrar el TDA Lista de Listas, a saber (como mínimo):

agregarUsuario()
buscarUsuarioPorId()
mostrarUsuarios() // muestra toda la lista de listas, cada usuario con su playlist
agregarPeliculaAUsuario() // agrega una Película al Usuario correspondiente
BorrarTodaLaLDL() // esta función borra toda la lista de listas, dejando la memoria libre para luego volver a trabajar

Dentro del manejo de la LDL se incluye el tema del manejo de la Playlist de cada usuario.

Queda a criterio de cada grupo elegir el método más conveniente para la persistencia en archivos de las Playlist, y para su “bajada” del archivo correspondiente a la LDL (a los fines de evitar ocupar espacios de memoria con información repetida, situación que se produciría si cada pelicula es guardada en forma

repetida dentro del archivo de peliculas que forma el banco de películas del sistema y además dentro de la struct de cada usuario que la haya elegido para su playlist).
Pueden optar por manejar un campo adicional en la struct Usuario, que sería un arreglo de enteros en el cual se guardan los id de las películas de la Playlist, y al momento de generar la LDL cada id de dicho arreglo es buscado en el arbol de películas para generar la lista de películas.
También pueden optar por crear una struct adicional que guardaría a modo de “parejas” el id de un usuario con el id de una película (cada usuario puede tener muchas películas, y cada película haber sido elegida por muchos usuarios). Esas struct se guardarían en otro archivo adicional, y para generar la LDL se recurriría a este archivo.
Ej:

```
typedef struct
{
    int idRegistroPlaylist;
    int idUsuario;
    int idPelicula;
} stRegistroPlaylist;
```

Esta estructura da forma al archivo de playlist de cada usuario. En cada registro se almacena el id del usuario, el id de la película y un id autoincremental para contabilizar los registros. A partir de esta información, se carga el arreglo de listas, buscando los datos del usuario en el archivo de usuarios y los datos de la película en el árbol de películas. Asimismo, deberá desarrollar las funciones necesarias para hacer el trabajo inverso (a partir de la LDL que se va cargando y actualizando en memoria, realizar la persistencia de los datos en los archivos correspondientes).

Main() y menúes: [10 puntos]

El sistema deberá contar con una presentación amigable con el usuario, construir menús de acceso a las diferentes estructuras y funcionalidades del sistema, y **de manera directa o indirecta, permitir probar todas las funciones desarrolladas**.

El desarrollo del sistema deberá ser ordenado, **identificando con comentarios cada una de las funciones realizadas, explicando brevemente lo que realizan**. **Se tendrá en cuenta, al momento de evaluar, la prolijidad del código y la organización de los módulos. Se recomienda agrupar los mismos por funcionalidad.**

El sistema tendrá que proporcionar el acceso a las diferentes funcionalidades mínimas, aunque se deja a libre desarrollo del grupo la forma de construir los menús:

Menú principal

1. Ingreso con User y Pass para Administradores
2. Ingreso con User y Pass para Usuarios
3. Registrarse

1- Ingreso con User y Pass Solo administradores:

Esta pantalla pide que se ingrese Usuario y Contraseña. Comprueba 1ro que el usuario exista. Si el usuario no existe, muestra mensaje. Si existe, comprueba la contraseña. Si es correcta, ingresa al sistema, si no lo es muestra mensaje.

Sub-Menú Administración de Usuarios:

- **Alta**: Una vez completado el formulario de alta se valida que no exista el usuario. Si no existe, se guardan los datos en los archivos correspondientes.
- **Baja**: Modificar el campo eliminado en la estructura y guardarlo.
- **Modificación**: En modificación se ingresa el nro de usuario. Si no existe, se muestra mensaje de error; si existe lo muestra y se ve una forma de poder modificar los campos. Ej: mostrar los campos con un número de orden y pedir el ingreso del nro de campo a modificar.
- **Consulta**: Para la consulta la metodología sería similar a listados.
- **Listados**: Para los listados se abre el archivo y se carga en una lista ordenada por nombre de usuario antes de mostrarlo.

Sub-Menu Administración de Películas:

- **Alta**: Una vez completado el formulario de alta se persiste la película en el archivo.
- **Baja**: Modificar el campo eliminado en la estructura y guardarlo.
- **Modificación**: En modificación se ingresa el nro de película. Si no existe, se muestra mensaje de error; si existe se ve una forma de poder modificar los campos. Por ejemplo: mostrar los campos con un número de orden y pedir el ingreso del nro de campo a modificar.
- **Consulta**: Para la consulta (búsqueda de una película en particular), la búsqueda se hace por id en el árbol de películas.
- **Listados**: Para los listados se recorre el árbol, y antes de mostrarlas, se cargan en listas ordenadas según los siguientes criterios:
    - por orden alfabético de acuerdo al nombre de la película
    - por antigüedad (desde la más nueva a la más vieja).
    - listar todas las películas correspondientes a determinado género recibido por parámetro.

2- Ingreso Con User y Pass para Usuarios Comunes:

Esta pantalla pide que se ingrese Usuario y Contraseña. Comprueba 1ro que el usuario exista. Si el usuario no existe, muestra mensaje. Si existe, comprueba la contraseña. Si es correcta, ingresa al sistema, si no lo es muestra mensaje.

Sub-Menú de Usuario

- **Ver perfil:** Muestra la información completa del usuario logueado.
- **Modificar datos:** Permite modificar los datos del perfil del usuario.
- **Baja**: Modificar el campo eliminado en la estructura y guardarlo.

Sub-Menú Películas

- **Mostrar películas de la plataforma:** Muestra el catálogo completo de películas de la plataforma. Para los listados se recorre el árbol, y antes de mostrarlas, se cargan en listas ordenadas según los siguientes criterios:
    - por orden alfabético de acuerdo al nombre de la película
    - por antigüedad (desde la más nueva a la más vieja).

        - listar todas las películas correspondientes a determinado género recibido por parámetro.
    - **Agregar películas a la Playlist:** A los fines de que el usuario pueda agregar películas a su lista, se le darán las siguientes opciones:
        - ver la lista completa de películas por orden alfabético (llamar a las funciones de los incisos anteriores)
        - ver la lista completa de películas por orden de antigüedad (llamar a las funciones de los incisos anteriores)
        - ver el listado de películas disponibles de un género en particular (llamar a la función de los incisos anteriores)
        - buscar una película en particular por su nombre
        - ir al sector de películas recomendadas

      Una vez elegida por el usuario la película a agregar, se busca la misma por id en el árbol de películas, y se agrega en la playlist del usuario (se agrega en la LDL, y antes de desloguearse o de cerrar el sistema se persiste en los archivos correspondientes)
    - **Eliminar películas de la Playlist:** Una vez elegida por el usuario la película a eliminar, se elimina la misma de la LDL, y antes de desloguearse o de cerrar el sistema se persiste la eliminación en los archivos correspondientes.
    - **Mostrar películas de la Playlist.**
    - **Películas recomendadas:** Muestra un listado de las películas recomendadas para el usuario (muestra todas las películas correspondientes al género que más se repite en su playlist).

3. **Registrarse:**
   Redirige al alta de usuario.