|  |  |  |
| --- | --- | --- |
| ***UTN – FRMDP Mar del Plata***  ***TSP - Laboratorio 2***  ***Trabajo Práctico Final***  *Octubre 2018* | ***Integrantes del grupo***  ***Latorre, Ariana Guillermina***  ***Fanjul,Rodrigo***  ***Burgueño, Jeremias*** | ***Nota*** |

#### **Introducción**

Con el propósito principal de integrar contenidos de otras asignaturas y además integrar todo lo aprendido en la materia laboratorio 2, sumando lo trabajado en laboratorio 1 hemos planteado la siguiente problemática:

* Reutilizar el sistema de Logueo desarrollado en el primer cuatrimestre, que gestiona la estructura usuario, guarda, recupera y encripta la contraseña del mismo en un archivo binario.
* Administrar un archivo de Películas (Alta, Baja, Modificación, Consulta y Listados)
* Administrar las películas vistas por cada usuario en un archivo independiente.

#### **Fundamentación**

El valor pedagógico de la propuesta se apoya en el aprendizaje colaborativo (se formarán grupos de 2 o 3 alumnos) y la integración de contenidos de otras asignaturas 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: **Según planificación de cada comisión**
* Si el sistema está codificado en su totalidad y funciona correctamente, se considerará aprobado con una nota mínima de 5.
* Si el sistema no está codificado en su totalidad (como mínimo un 50 % 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.
* Es obligatorio la presentación de este trabajo para aprobar la materia.

**Pautas Generales**

Utilizando el sistema desarrollado en la cátedra 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, sumando una nueva manera de persistir los datos de las películas vistas por cada usuario.

El sistema deberá permitir gestión de Usuarios, Películas y Películas vistas (***UTN-etflix***) persistiendo la información en archivos binarios. Las claves de acceso de los usuarios deberán guardarse encriptadas en el archivo, tarea que ya fue desarrollada en el primer cuatrimestre.

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](http://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.

Para la persistencia de datos en los archivos (“usuarios.dat”, “peliculas.dat” y “peliculasVistas.dat”) utilizaremos las siguientes estructuras de datos:



Los registros del archivo binario “***peliculasVistas”***, se construyen a partir de los datos más representativos del Usuario (idUsuario), de la Pelicula (idPelicula) y de un idPeliVista (campo autoincremental) para identificar el número de registro. Se establece una relación de 1 a N entre los archivos, donde un Usuario puede haber visto muchas películas y una película puede haber sido vista muchas veces.

**Detalle de estructuras y funcionalidad básicas:**

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

**Estructura de Película**

|  |  |
| --- | --- |
| typedef struct {  int idPelicula;  char nombrePelicula[60];  char director[30];  char genero[20];  char pais[20];  int anio;  int valoracion;  int pm;  int eliminado;  } stPelicula;  Importar y adaptar las funciones de Alta, Baja, Modificación, Consulta y Listados de Películas que ya fueron desarrollados en el TP Final de Laboratorio 1.  **Estructura de Usuario**   |  | | --- | | typedef struct {  int admin;   int idUsuario;  char nombreUsuario[15];  char genero;  char pais[20];  int pass[2][5];  int anioNacimiento;  int eliminado;  } stUsuario; |   Importar y adaptar las funciones de Alta, Baja, Modificación, Consulta y Listados de Usuarios que ya fueron desarrollados en el TP Final de Laboratorio 1, con el correspondiente manejo del password encriptado.  **Lista de Películas (lista simple) [ 15 puntos ]**  typedef struct {  stPelicula p;  struct nodoListaPelicula \* sig; } nodoListaPelicula;  Deberán codificar todas las funciones necesarias para administrar el TDA Lista Simple, a saber (como mínimo):  inicLista()  crearNodoLista()  agregarAlPrincipio()  agregarAlFinal()  agregarEnOrdenPorNombreDePelicula()  mostrarLista() // modularizar  borrarNodoPorIdPelicula() |

**Árbol de Películas [ 20 puntos ]**

|  |
| --- |
| typedef struct {  stPelicula p;  struct nodoArbolPelicula \* izq;  struct nodoArbolPelicula \* der; } nodoArbolPelicula; |

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

inicArbol ()

crearNodoArbol ()

insertarNodoArbol (ordenado por idPelicula)

mostrarArbol (son tres funciones, recorriendo inOrder, postOrder, preOrder) // modularizar

borrarUnNodoArbol (buscarlo por idPelicula)

buscarPelicula (por idPelicula)

**cargarArbolDesdeArchivo():** Al inicio del sistema, deberán cargar todas las películas del archivo, sobre un árbol binario ordenado por idPelicula, 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 idPelicula 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.

**Estructura de Arreglo de Usuarios (Arreglo de listas) [ 20 puntos ]**

|  |
| --- |
| typedef struct {  stUsuario usr;  nodoListaPelicula \* listaPelis; } stCelda; |

En el programa principal (main) se definirá un Arreglo de Listas de dimensión justa, utilizando la función malloc(), para lo cual deberán determinar la cantidad de usuarios activos en tiempo de ejecución.

Este arreglo se cargará de forma automática al iniciar el sistema, con todos los usuarios activos y sus películas vistas. El trabajo en el sistema se realizará sobre esta estructura y, por ejemplo, al momento de ver una película se realizarán las acciones necesarias para actualizar la información sobre el ADL. Luego, una vez que el usuario quiera desloguearse o antes de cerrar el programa, se persistirán estos los datos en los archivos.

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

agregarUsuario() // crea un nuevo usuario en el arreglo

buscarUsuario() // busca un usuario por idUsuario y retorna la posición que ocupa en el arreglo

mostrarUsuarios() // muestra todo el arreglo de listas, cada usuario con sus películas vistas

agregarPeliculaToUsuario() // agrega una Pelicula al Usuario correspondiente

limpiarArregloDeListas() // esta función “vacía” todo el arreglo de listas, dejando la estructura preparada para volver a trabajar

persistirPeliculasVistas() // esta función realizará la persistencia de todas las películas vistas en el archivo correspondiente

**Estructura de Películas vistas por cada Usuario [ 15 puntos ]**

|  |
| --- |
| typedef struct {  int idPeliVista;  int idUsuario;  int idPelicula; } stPelisVistas; |

Esta estructura da forma al archivo de películas vistas por 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 y los datos de la película en el árbol de películas. Para hacer esto, deberá desarrollar una serie de funciones que sean invocadas por la función **pasarDeArchivoPelisVistasToADL().**

Asimismo, deberá desarrollar las funciones necesarias para hacer el trabajo inverso. A partir del arreglo de listas que se va cargando y actualizando en memoria, realizar la persistencia de los datos en el archivo de películas vistas, recorriendo el ADL y tomando los datos allí almacenados.

**actualizarPeliculasVistas().**

**Codificar las funciones necesarias para persistir esta estructura en un archivo binario y las que necesite para la interacción con el sistema.**

**La función principal - Main() y menús: [ 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

**3.** Registrarse

**1- Ingreso con User y Pass Solo administradores:**

**Sub-Menú Administración Usuarios:**

* **Alta**: Una vez completado el formulario de alta se valida que no exista el usuario. Si no existe se encripta la contraseña (se convierte el pass en una matriz de 2xn se multiplica por la matriz testigo que la tengo como constante y el resultado queda en la matriz pass de la estructura) entonces 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 desencripta la contraseña y lo muestra; y se ve una forma de poder modificar los campos. Una forma sería mostrar los campos con un número de orden y pedir el ingreso del nro de camp a modificar.
* **Listados**: Se muestran los “usuarios activos” desde el Arreglo de Listas, mostrando sus datos y las películas vistas por cada uno.
* **Consulta**: Para la consulta la metodología sería similar a listados, buscando por id o por nombre de usuario.

**Sub-Menu Administracion 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 camp a modificar.
* **Listados**: Se muestra la información cargada en el árbol de películas, permitiendo recorrer el mismo en modo preorder, inorder y postorder. Se cargan en el árbol únicamente las películas activas.
* **Consulta**: Para la consulta la metodología sería similar a listados, mostrando una película en particular, por id o por nombre.
* Borrar una Película del árbol.

**2- Ingreso Con User y Pass:** Esta pantalla pide que se ingrese Usuario y Contraseña.

Se constata la existencia del usuario contra la información existente en el Arreglo de Listas, si el usuario existe comprueba que la contraseña sea correcta y muestra los datos del mismo. Si el usuario no existe << muestra mensaje >> y si el usuario existe pero la contraseña no es correcta << muestra mensaje >>

**Sub-Menú de login exitoso**

* **Ver perfil:** Muestra la información completa del usuario logueado.
* **Mostrar películas:** Muestra el catálogo de películas desde el árbol.
* **Ver pelicula:** Se ingresa el id de la película, se la busca en el árbol y se realizan las acciones necesarias para almacenar la misma en la lista correspondiente al usuario logueado.
* **Borrar una película** vista de la lista perteneciente al usuario logueado.
* **Películas recomendadas:** Muestra un listado de las películas recomendadas para el usuario en base a su historial de reproducción. Queda a criterio del equipo el algoritmo de recomendación *(pero esta vez en serio…. no como el cuatrimestre pasado, sorprendenos!!).*

**3. Registrarse:**

Redirige al alta de usuario.

-----------------------------------------------------------------------------------------------------------------------------------------

**Tabla de puntuación:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Obtenido** | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| **Nota** | 1 | 2 | 2 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **Observación** | *Desaprobado* | | | | *Aprobado* | | | | | |

**DIARIO DE TRABAJO**

**LINK** [**TRELLO**](https://trello.com/b/HeI4CpBO/burgue%C3%B1o-fanjul-latorre)

**24/10:**

Vinimos todos a clase, pero había partido de boca.

Empezamos a charlar las funciones del tp.

ANOTACIONES

(Guille)

-**Arreglo Pelis vistas**:cargar nuevamente en el archivo, y que la estructura reciba un arreglo dinámico, para que cada vez que se agregue una película vista el arreglo aumente en 1. Así siempre que se vuelva a cargar los usuarios al archivo sobreescrito se utilice la función malloc(sizeof(archivoDinamico)) .

**-Mostrar Películas:** que la interfaz sea paginada, con opción de volver atrás y página siguiente.

-**Recomendadas:** Al finalizar la película el user califica la película entre 1 y 5. Luego muestra las películas recomendadas en cuanto a la calificación de acuerdo a las películas del genero que mas haya visto este usuario, y por otra parte películas similares a la que termino de ver.. **Valoración**: En caso de que esa la primera que vea se muestran solo las similares.

-**SOBRESCRIBIR ARCHIVO:** levantar todo el archivo al iniciar el programa y trabajar toda su duración sobre los árboles y arreglos de listas de usuarios. Al finalizar sobrescribir el archivo con las películas o usuarios nuevos. De esta forma entra en juego la necesidad de un arreglo dinámico en la estructura de pelis vistas**(pensarlo).**

**25/10:**

Frase de hoy “Hoy es jueves y se programa fuerte” by Gustavo Sonvico.

Realizamos la división de codificado de TDAs:

TDA árbol películas (carioca);

TDA listas películas (jere);

ADL usuarios (guille);

ANOTACIONES

Jere: Crear un char con la url base tanto para películas como usuarios. Éste sería: <https://www.utn-eflix.com/peliculas/> y [https://www.utn-eflix.com/user/](https://www.utn-eflix.com/peliculas/) respectivamente y cuando se cargue una entrada nueva, agregar el nombre al final del link.

Hacer un encabezado que muestre el link en donde el usuario esté navegando.

**26/10:**

Finalizamos las funciones básicas del TDA tanto de árboles(Carioca) como de lista de películas(Jere).

Continuamos con el TDA del ADL de usuarios(Guille) y comenzamos con las funciones del struct de pelisVistas(Jere).

ANOTACIONES

(Jere)

Crear una función la cual cargue el archivo al arreglo de pelisVistas donde a la vez a través de la ID de la película y del usuario se recorre la lista y el arreglo respectivamente para ir cargando las películas a la lista dentro del struct de usuarios. Pude llegar a una solución aunque aun debo probarla (Ver en matriz de soluciones).

**1/11:**

GUILLE(NO) TUVO UNA IDEA

Para cargar el árbol equitativamente propongo:

Cuando se cargue el árbol, nos ubicamos en el medio (quedó ahí).

**2/11:**

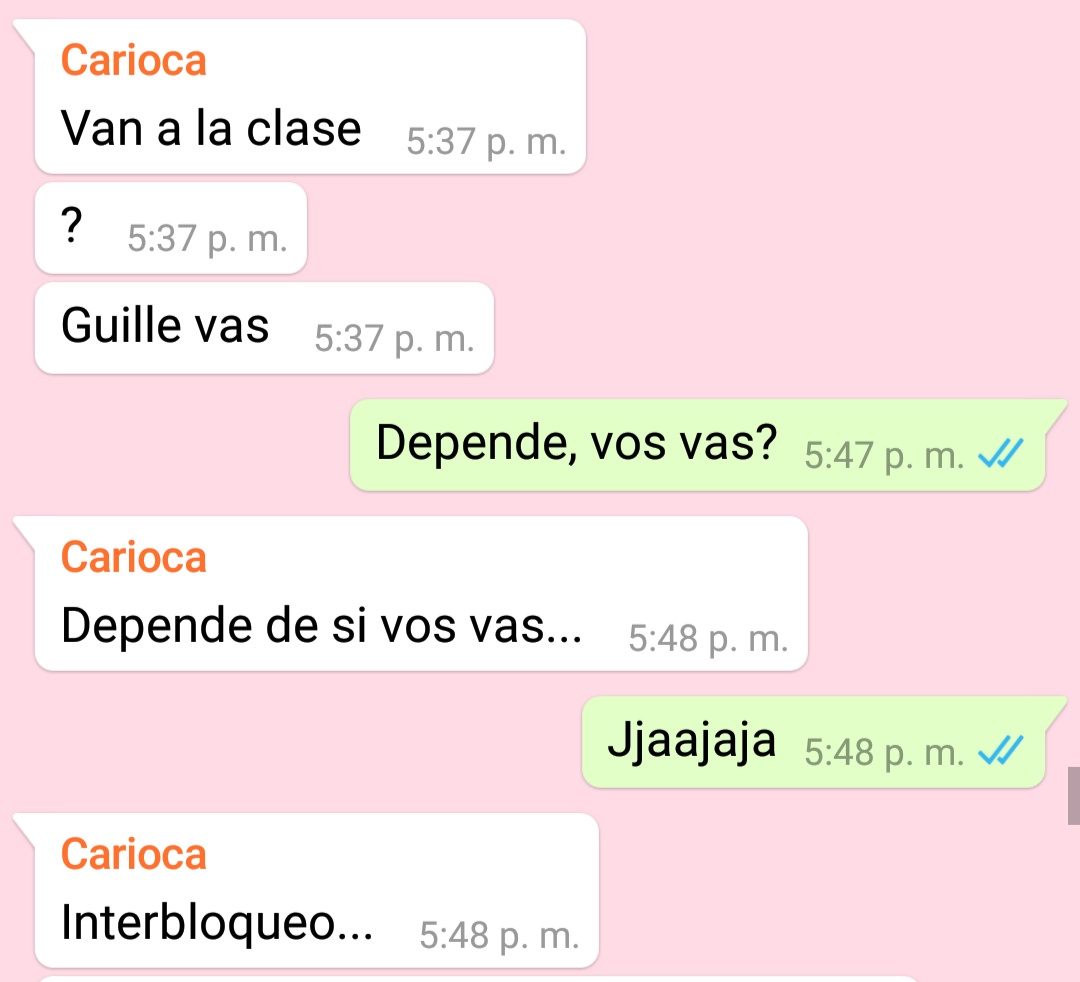
Unimos todos los TDA y compiló(Carioca-Jere-Guille). Tuvimos problemas con las variables globales relacionadas a los archivos.

**03/11 :**

Problemas con las variables globales solucionados con el uso de include guards (Ver matriz de soluciones). Ahora adaptando registro para funcionar con arreglo de listas(Carioca).

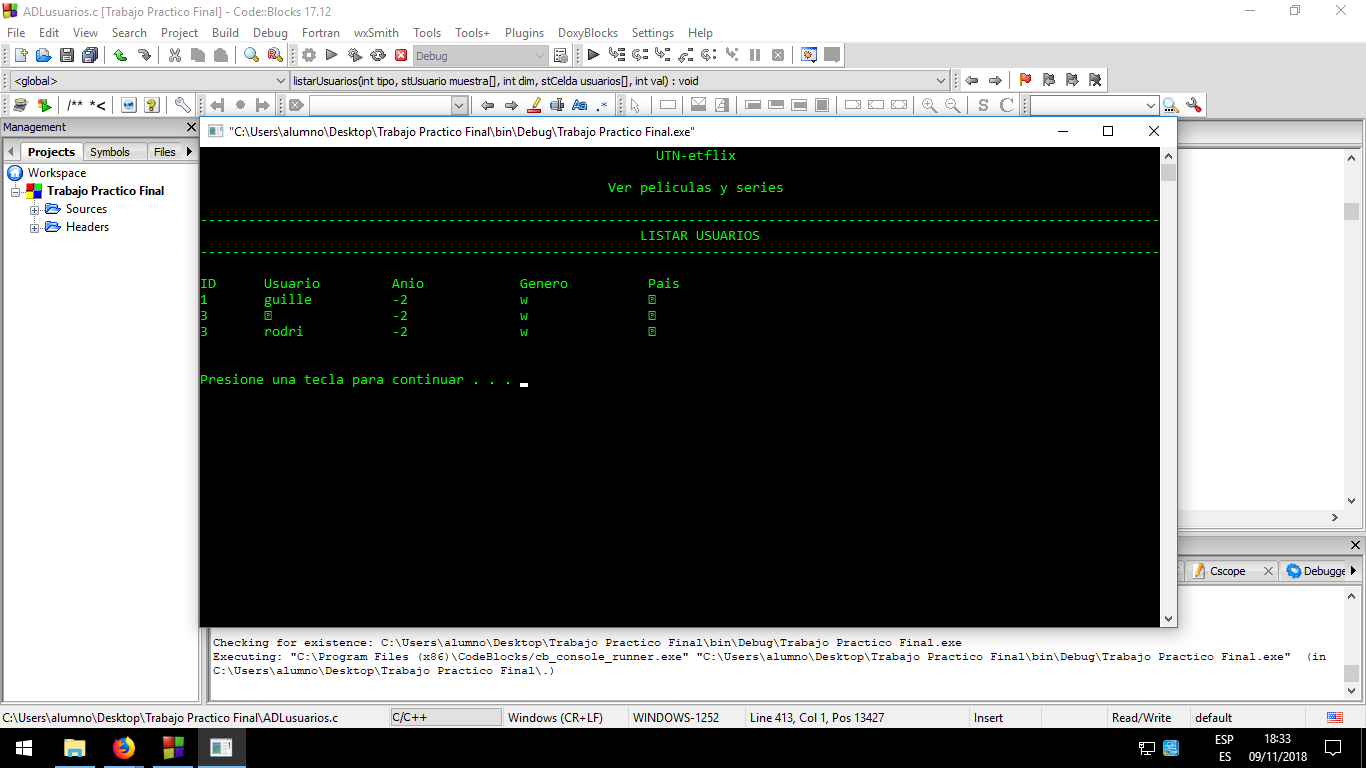
**08/11 :**

Jere estaba enfermo y Carioca y Guille…



Interbloqueo: proviene de haber rendido parcial de Arquitectura de Sistemas Operativos el día anterior.

De todas formas trabajamos en remoto, solucionamos un problema. Consistia en que al entrar en modo administrador y listar los usuarios, al segundo usuario no lo mostraba, ni tampoco estaba cargado en el ADL, y pisaba la ID de usuario con la última ID cargada.

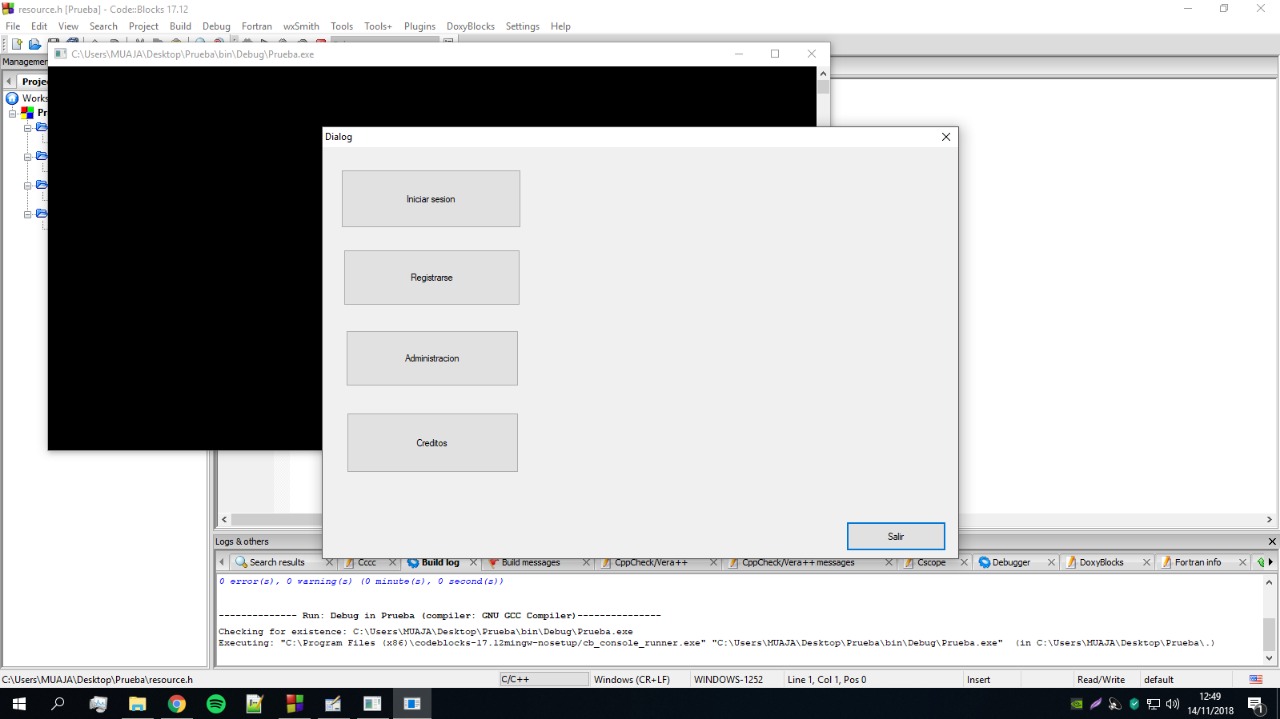


(Ver en matriz de soluciones)

**09/11 :**

Jere sigue enfermo(pero trabajando) , mientras tanto Carioca y Guille se pelean por la división de tareas.

Proponemos cambiar la interfaz, primero investigaremos GTK, si no lo entendemos usamos otro.



Intentamos pero había que cambiar la composición de menús y funciones.

ANOTACIONES

(Guille)

Para cargar el árbol:

Pasar el archivo a un arreglo. Árbol más de 1 película.

Busco la id de los dos extremos y la id del medio.

a=id menor b=id medio c=id mayor

/// rama izquierda

int conta=1;contb=1; int mayor;

if(a+cont >= b-cont)

{

if(contb == 1)

{

mayor = b-cont;

contb++;

acomoda resultado de b-cont;

}

else

{

acomoda resultado de b-cont;

}

acomoda el resultado de a;

conta++;

contb++;

}

else

{

if(a+cont == b-cont)

{

acomoda el resultado de a;

}

acomoda el valor de a;

}

/// rama derecha

int cont=1;

if (b+cont >= c-cont)

{

acomoda el resultado de b y resultado de c;

cont ++;

}

else

{

if(b+cont == c-cont)

{

acomoda el resultado de c;

}

acomoda el valor de c;

}

También hablamos sobre acceder a vídeos (trailers de peliculas) de youtube mediante links anclados al código(Carioca).

**10/11:**

Se suspendió el partido, Jere sigue enfermo y contagió a Guille. Carioca se queja de que tiene el equipo en terapia intensiva.

Logramos balancear el árbol. El pseudocódigo de Guille se aplicó perfectamente en la papelera de reciclaje y Carioca logró una solución mucho más simple (Ver en matriz de soluciones).

**11/11:**

La madrugada del domingo asaltaron y golpearon a los abuelos de Guille.

Después de una mañana de apoyo en el grupo, continuamos con el trabajo.

Implementamos GitHub para la actualización del tp, y comenzamos a darle forma al tp escrito.

**12/11-14/11:**

Durante dos días nos dedicamos a estudiar estadística, que rendimos el lunes, y programación II, que rendimos el miércoles.

**15/11:**

**DIA DE ENTREGA**

Llegó el día de entregar, terminamos todos los incisos del tp escrito y del práctico. Queda unir algunas cosas, pero confiamos en que llegamos a entregarlo hoy.

Falta agregar películas recomendadas y agregarlo a la matriz de soluciones.

**MATRIZ DE SOLUCIONES**

**TDA en diferentes librerías:**

Nuestro programa tiene una estructura piramidal, en la que una sola librería de estructuras contiene todos los tipos de datos con los que vamos a trabajar a lo largo de todo el desarrollo de la aplicación. Esa librería debe ser incluída en todos los TDA que la utilizan.

El problema consiste en que esa librería se incluía más de una vez en el main porque estaba presente en todos los TDA utilizados por el programa.

Logramos solucionar el problema por medio de la utilización de include guards (o macro guards).

|  |
| --- |
| **ESTRUCTURAS.h** |
| #ifndef ESTRUCTURAS\_H\_INCLUDED  #define ESTRUCTURAS\_H\_INCLUDED  /\* Código estructuras \*/  #endif |

Este macro le indica al compilador que tiene que incluir esa parte del código una sola vez a la hora de compilar y nos permite utilizar todas las librerías que queramos sin obtener el error de doble declaración que teníamos previo a su utilización.

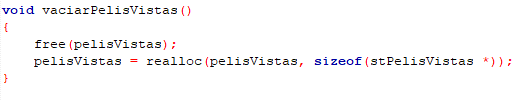
**Pelis Vistas:**

Crear una función la cual cargue el archivo al arreglo de pelisVistas donde a la vez a través de la ID de la película y del usuario se recorrer la lista y el arreglo respectivamente para ir cargando las películas a la lista dentro del struct de usuarios.

Se resolvió modularizando (haciendo muchas funciones) que se encarguen de cada paso necesario para lograr el objetivo.

Primero la función que va leyendo el archivo y cargando los datos. Primero a un arreglo de pelisVistas que se va reacondicionando a medida que entren más datos a la vez que van agrandando los válidos. A la vez, esto llama una función que buscando las idUser se van copiando los datos de las películas al usuario que corresponde.

A su vez, cada vez que se terminan de guardar las pelisVistas en el archivo, el arreglo queda cargado con datos que ya no necesitamos. Se soluciono creando una función que vacíe el arreglo utilizando free(), y vuelva a redimensionar el tamaño a vacío, sizeof(stPelisVistas \*).



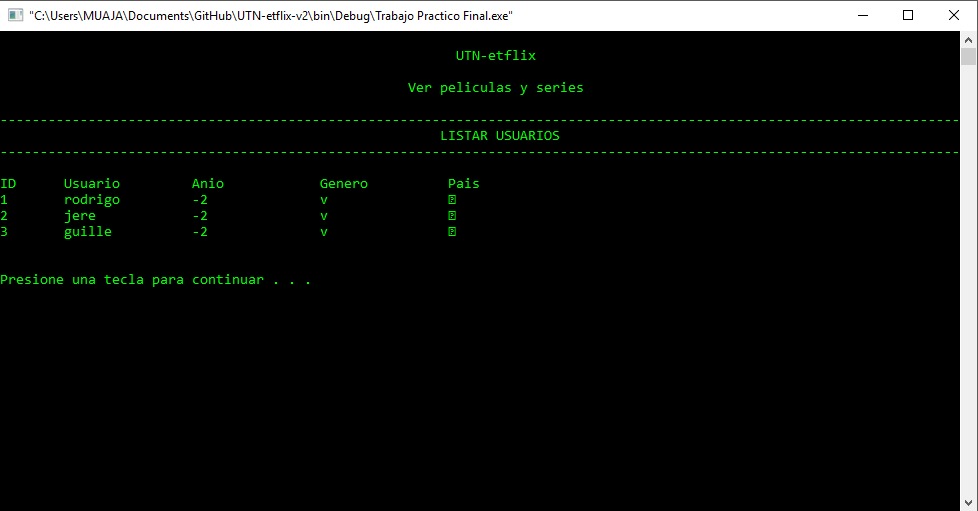
**Al entrar en modo administrador y listar los usuarios, al segundo usuario no lo mostraba, ni tampoco estaba cargado en el ADL, y pisaba la ID de usuario con la última ID cargada:**

Fueron tres problemas que teníamos. El problema más grave fue que en el header de estructuras y estábamos declarando el ADL de usuarios de celdas e inicializando en 1, cuando no tendría que ir inicializado porque no hay ningún usuario. Para solucionarlo declaramos el ADL de usuarios como un puntero sin inicializar, con cero celdas.

Después, en la función de cargado del mismo, lo inicializa si hay más de un usuario, si no hay más de uno no hace falta inicializarlo, lo deja en cero.

El segundo problema que teníamos era que en la función de redimensionar el ADL, le estábamos pasando una copia el ADL original, entonces cuando lo redimensionabamos no retornaba el nuevo arreglo y perdíamos la dirección de memoria con todos los datos.

El tercer problema que teníamos era que todas las funciones del ADL recibían una copia del mismo, en vez de usar el arreglo global. Cambiamos los parámetros de esas funciones, eliminando el campo arreglo, entonces así trabajan dentro de las mismas con el arreglo global del programa. Así cuando, por ejemplo, se modifica un usuario también se modifica el ADL global que trabaja con otras funciones.



**Desordenar el Archivo:**

El algoritmo lo que hace es mientras vas cargando el arbol, lo vas balanceando recursivamente.

Nosotros vamos comprobando las alturas del subárbol izquierdo y del subárbol derecho de cada nodoArbol a medida que vamos cargando el árbol. Si el subárbol derecho es más alto que el izquierdo, rotamos hacia la izquierda, y si no al revés, para la derecha. Eso hace que aumente en uno y se reste en uno la diferencia de la altura. Podemos ajustar hasta dos niveles de altura por cada vez que comprobamos.

Entonces lo que hace es:

-Empieza a cargar, el primero lo carga como raíz

-El segundo lo carga para la derecha

-El tercero entra en la condición: ¿La altura del subárbol izquierdo es dos veces más alto que el subárbol derecho? Es verdadero, la condición se cumple.

-Hace una rotación a la izquierda:

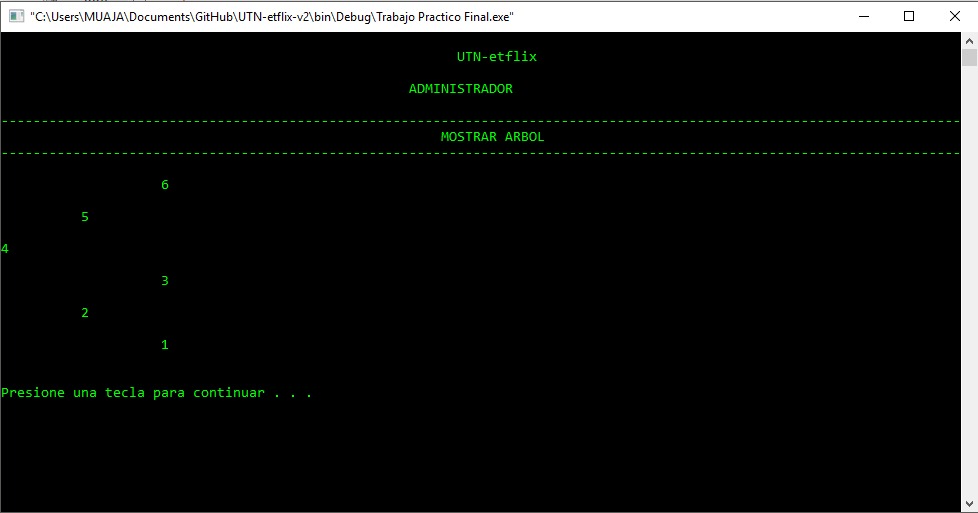
-el nodo raíz pasa a ser el subárbol izquierdo;

-el primero del subárbol derecho pasa a ser la raíz;

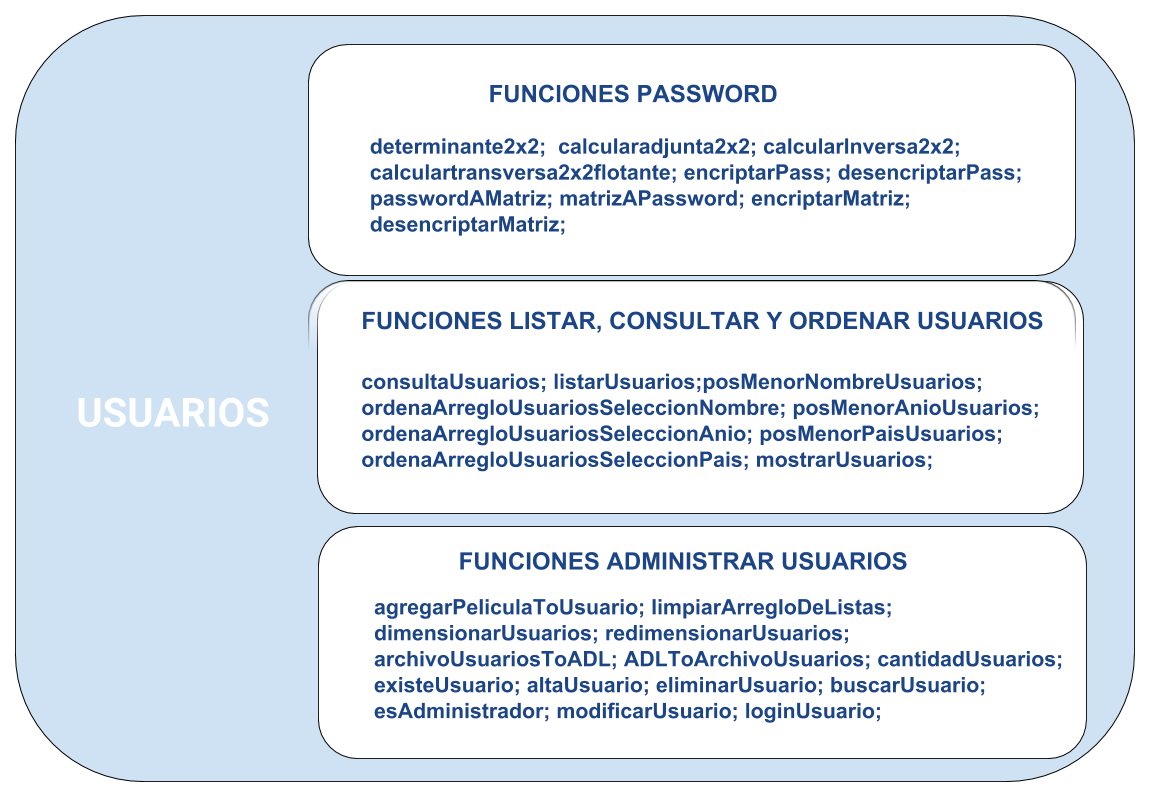
-y el tercero que cargamos pasa a ser el subárbol derecho.

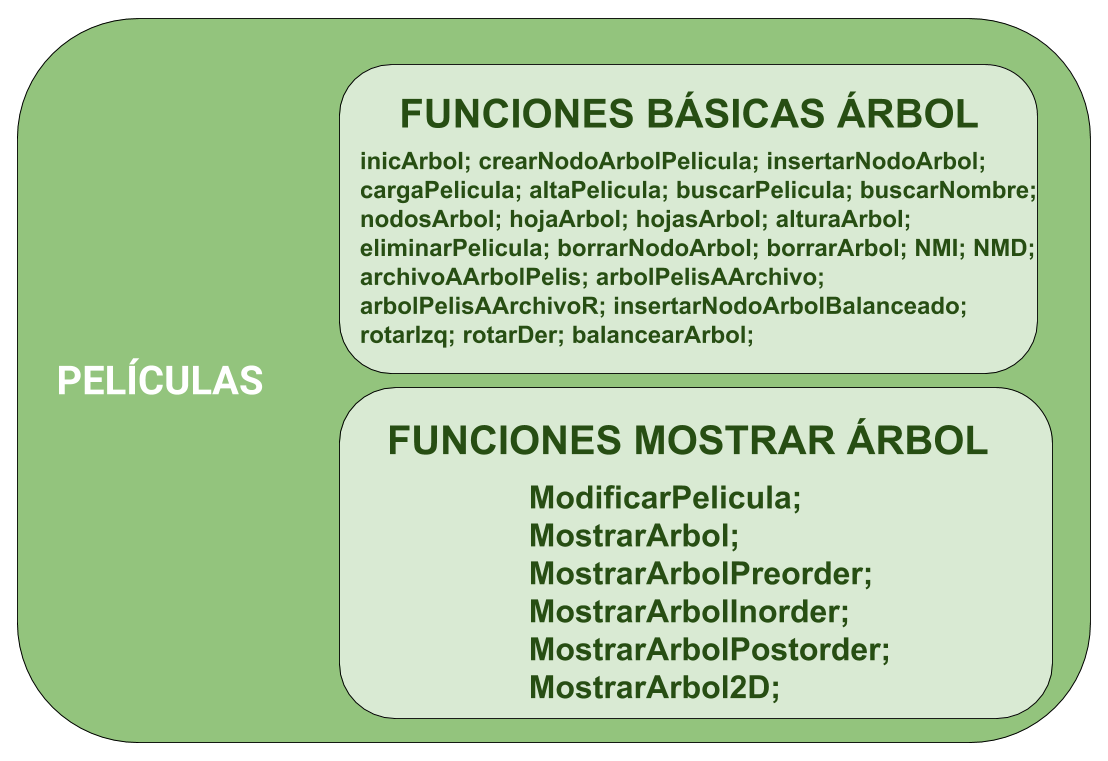
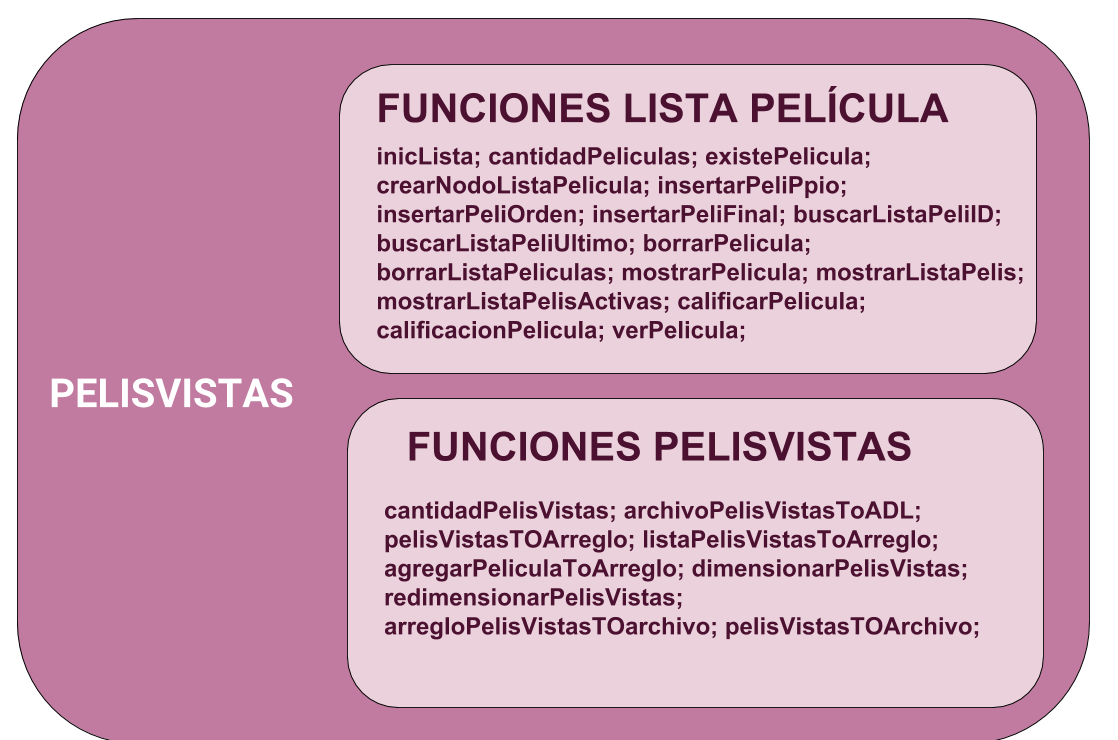
-Carga uno más y no hay ningún problema porque la altura del subárbol izquierdo varia solo en uno a la del subárbol derecho.

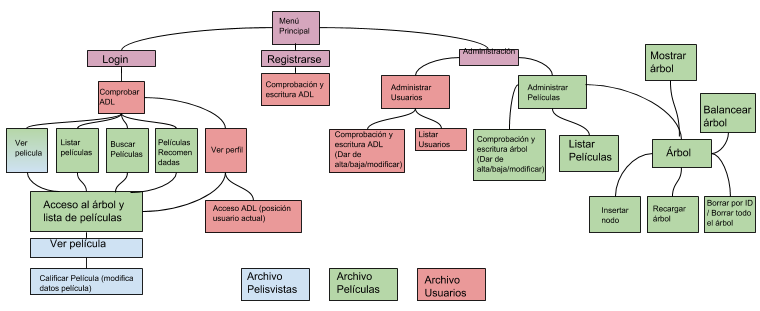
-Carga otro más y vuelve a hacer otra rotación.



**DIAGRAMA DE ESTRUCTURAS**

****

****

****

**MANUAL DE USUARIO**

Bienvenido al manual del usuario de UTN-etflix.

Este manual lo ayudará a guiarse en el uso de nuestro programa. Valoramos que no haya elegido.

**MENÚ INICIO**

Ni bien abra el programa, ésta será la primera pantalla que verá. En ella le aparecerán 4 opciones:

1. Login de Usuario.
2. Registrarse
3. Administración.
4. Ver créditos.

0- Salir a Windows.

**Login de Usuario**: Si ya está registrado, allí es donde usted ingresa los datos de su cuenta para acceder al programa.

**Registrarse:** Es donde podrá crear una nueva cuenta.

**Administración:** Es a donde, si usted es administrador, ingresa para poder modificar lo que desee.

**Ver créditos**: Lo mandará a una página web donde verá los creadores del programa.

**Salir a Windows:** Cerrar el programa.

Para acceder a cada función debe ingresar el número que está a la izquierda de cada opción.

**LOGIN USUARIO**

Se accede si usted ingresó 1 en el menú principal. Aquí se le pedirá que ingrese su nombre de usuario y si, usted ya tiene una cuenta, la contraseña.

**MENÚ PRINCIPAL**

Luego de acceder a su cuenta exitosamente, le aparecerán las siguientes opciones:

1. Ver película.
2. Listar películas.
3. Buscar película.
4. Películas recomendadas.
5. Ver perfil.

0- Salir al menú inicio.

**Ver película:** Se le mostrará una lista para ver todas las películas de nuestra videoteca e ingresando el número de ID de la misma (se muestra junto con la película) usted podrá verla. Luego se le mandará a un menú donde podrá: Calificar la película, verla de nuevo o buscar otra película distinta para ver.

**Listar película**: Se le mostrarán 7 maneras diferentes para buscar películas de forma más sencilla:

1. Por título.
2. Por género.
3. Por país.
4. Por valoración.
5. Por año.
6. Por clasificación.
7. Ver todas las películas.

0- Volver al menú principal.

**Buscar película:** Podrá buscar la película ingresando el nombre de la misma.

**Películas recomendadas:** Se le mostrarán las películas recomendadas según sus preferencias.

**Ver perfil:** Podrá ver sus datos y modificarlos.

**Salir al menú inicio**: Vuelve al menú anterior.

**REGISTRO DE USUARIO:**

Se accede si usted ingresó 2 en el menú principal. Aquí es donde se le pedirán los datos necesarios para registrarse.

-Nombre de usuario.

-Contraseña.

-Año de nacimiento (debe estar entre 1900 y 2018).

-Genero (m para masculino y f para femenino).

-Nacionalidad.

**ADMINISTRACIÓN:**

Se accede si usted ingresó 3 en el menú principal. Aquí es donde, si usted es administrador, podrá manejar la base de datos.

En primera instancia se le pedirá que ingrese su nombre de usuario y contraseña. Esta se validará si es o no administrador.

**MENÚ ADMINISTRACIÓN**

Se le mostrarán 3 opciones:

1. Administrar usuarios.
2. Administrar películas.

0 - Volver al inicio.

**Administrar usuarios:**

Tendrá las siguientes 4 opciones:

1-**Dar de alta a un usuario:** Podrá volver a activar la cuenta de un usuario eliminado.

2-**Dar de baja a un usuario**: Podrá desactivar la cuenta de un usuario.

3-**Modificar los datos de un usuario**: Se le mostrarán los campos para que pueda modificar el que desee.

4-**Buscar usuario:** Podrá buscar el usuario. Le aparecen las siguientes opciones:

Buscar por nombre; Por año; Por país; Todos.

5- **Listar usuarios:** Le mostrará un menú para buscar al usuario deseado.

Dicho menú contendrá las opciones de: Buscar por nombre; Por año; Por país;

Todos.

0- **Volver hacia atrás:** Te devuelve al menú anterior.

**Administración Películas:**

Tendrá las siguientes 7 opciones:

1-**Dar de alta una película:** Podrá volver a hacer visible para los usuarios una película.

2-**Dar de baja una película**: Podrá hacer que una película deje de estar disponible para los usuarios.

3-**Modificar los datos de una película:** Se le mostrarán los campos para que pueda modificar el que desee.

4**-Buscar Película**: Podrá buscar la película. Le aparecen las siguientes opciones: Buscar por título; Por género; Por país; Por valoración; Por año; Por clasificación; Todas.

5-**Listar películas:** Le mostrará un menú para buscar la película deseada. Dicho menú contendrá las opciones de: Buscar por título; Por género; Por país; Por valoración; Por año; Por clasificación; Todas.

6-**Árbol de películas**: El administrador podrá acceder al árbol que administra las películas. Dentro de dicho menú se podrá:

1-**Insertar nodo en árbol:** Podrá insertar un nuevo nodo (con los datos de una película) dentro del árbol.

2-**Mostrar árbol:** Se le mostrará al usuario el árbol con las siguientes opciones: Preorder; Inorder; Postorder; 2D.

3-**Borrar nodo por ID:** El administrador contará con la posibilidad de eliminar un nodo conociendo la ID de la película correspondiente.

4-**Borrar todo el árbol:** Se borrará todo el árbol.

5-**Recargar árbol:** Se recargará el árbol por si ocurre el problema de que está vacío.

6-**Balancear árbol:** Al agregar un nuevo nodo, el administrador con esta opción podrá mantener el equilibrio del árbol.

0-**Volver hacia atrás**: Te devuelve al menú anterior.