UTN – FRMDP Mar del Plata ***TUP - Laboratorio 2*** ***Trabajo Práctico Final*** *Octubre 2020*	***Integrantes del grupo***	***Nota***

Introducción

Con el propósito principal de integrar contenidos aprendidos en la materia laboratorio 2, sumando lo trabajado en laboratorio 1 hemos planteado la siguiente problemática:

- Codificar un sistema de Logueo que gestione la estructura **Clientes.**
- Administrar un archivo de **Productos** (Alta, Baja, Modificación, Consulta y Listados)
- Administrar un **Pedido** por cada cliente 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)** 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
- Árboles binarios.
- Pilas y Filas implementadas con Listas (simples o dobles).
- 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 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.
- 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 los productos encargados por cada cliente.

El sistema deberá permitir gestión de Clientes, Productos y Pedidos (*YouTicsYa*) persistiendo la información en archivos binarios.

Como metodología de trabajo, se requiere crear una documento de texto (o carpeta) 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:

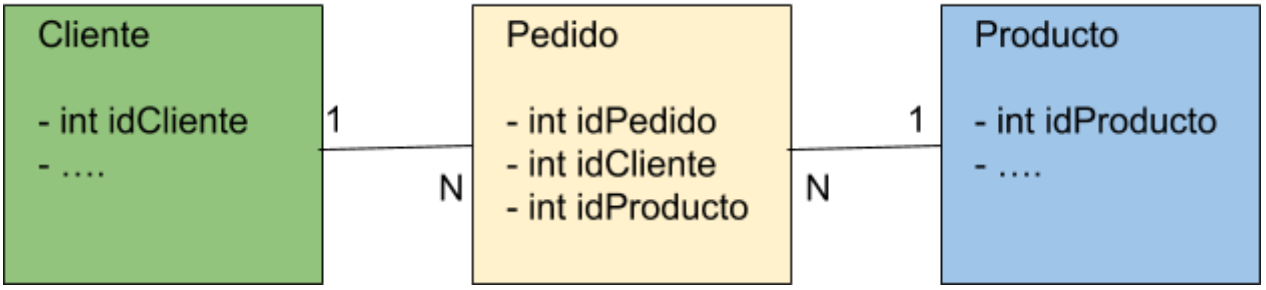
Informe final - Documentación a entregar: [10 puntos]

- 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.

Al momento de efectuar la entrega del proyecto, deberán agregar todo el material digital solicitado precedentemente.

Para la persistencia de datos en los archivos (“clientes.dat”, “productos.dat” y “pedidos.dat”) utilizaremos las siguientes estructuras de datos:



Los registros del archivo binario “***pedidos”***, se construyen a partir de los datos más representativos del Cliente (idCliente), del Producto (idProducto) y de un idPedido (campo autoincremental) para identificar el número de registro. Se establece una relación de 1 a N entre los archivos, donde un Cliente puede haber solicitado muchos productos y un producto puede haber sido solicitado muchas veces.

Detalle de estructuras y funcionalidad básicas:

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

Estructura de Producto

```
typedef struct {
   int idProducto;
   char nombre[30];
   char marca[20];
   float precio;
   int eliminado;  // indica 1 o 0 si el producto fue eliminado
} stProducto;
```

Estructura de Cliente

```
typedef struct
{
    int idCliente;
    char nombre[30];
    char apellido[30];
    char userName[20];
    char password[20];
    char mail[30];
    char domicilio[30];
    char genero;
    int rol;          // 1: es admin - 0: cliente
    int eliminado;   // indica 1 o 0 si el cliente fue eliminado
} stCliente;
```

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

Lista de Productos (lista simple) [15 puntos]

```
typedef struct
{
    stProducto p;
    struct nodoListaProducto * sig;
} nodoListaProducto;
```

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

inicLista()

crearNodoLista()
agregarAlPrincipio()
agregarAlFinal()
agregarEnOrdenPorNombreDeProducto()
mostrarLista() // modularizar
borrarNodoPorIdProducto()

Árbol de Productos [20 puntos]

```
typedef struct
{
    stProducto p;
    struct nodoArbolstProducto * izq;
    struct nodoArbolstProducto * der;
} nodoArbolstProducto;
```

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

inicArbol ()
crearNodoArbol ()
insertarNodoArbol (ordenado por idProducto)
mostrarArbol (son tres funciones, recorriendo inOrder, postOrder, preOrder) // modularizar
borrarUnNodoArbol (buscarlo por idProducto)
buscarProducto (por idProducto)

cargarArbolDesdeArchivo(): Al inicio del sistema, deberán cargar todos los Productos del archivo, sobre un árbol binario ordenado por idProducto, de forma tal que las búsquedas se realicen de forma más eficiente.

Tenga en cuenta que, seguramente, su archivo de productos está ordenado de forma creciente por idProducto 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 Clientes (Arreglo de listas) [20 puntos]

```
typedef struct
{
    stCliente cliente;
    nodoListaProducto * listaDeProductos;
    float costoTotalDelPedido;
} 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 clientes activos en tiempo de ejecución.

Este arreglo se cargará de forma automática al iniciar el sistema, con todos los clientes activos y sus productos solicitados. El trabajo en el sistema se realizará sobre esta estructura y, por ejemplo, al momento de agregar un producto al pedido, se realizarán las acciones necesarias para actualizar la información sobre el ADL. Luego, una vez que el cliente 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 Arreglo de Listas, a saber (como mínimo):

agregarCliente() // crea un nuevo cliente en el arreglo
buscarCliente() // busca un cliente por idCliente y retorna la posición que ocupa en el arreglo
mostrarClientes() // muestra todo el arreglo de listas, cada cliente con su pedido
agregarProductoToCliente() // agrega un Producto al Cliente correspondiente
limpiarArregloDeListas() // esta función "vacía" todo el arreglo de listas, dejando la estructura preparada para volver a trabajar
persistirPedidos() // esta función realizará la persistencia de todas los pedidos en el archivo correspondiente

Estructura de pedidos por cada Cliente [15 puntos]

```
typedef struct
{
    int idPedido;
    int idCliente;
    int idProducto;
} stPedido;
```

Esta estructura da forma al archivo de pedidos de cada cliente, en cada registro se almacena el id del cliente, el id del producto y un id autoincremental para contabilizar los registros.

A partir de esta información, se carga el arreglo de listas, buscando los datos del cliente en el archivo y los datos del producto en el árbol de productos. Para hacer esto, deberá desarrollar una serie de funciones que sean invocadas por la función **pasarDeArchivoPedidosToADL().**

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 pedidos recorriendo el ADL y tomando los datos allí almacenados. La función se llamará **actualizarPedidos().**

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]

IMPORTANTE: LA NAVEGABILIDAD DEL PROGRAMA.

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 clientes
3. Registrarse

1- Ingreso con User y Pass Solo administradores:

Sub-Menú Administración Clientes:

- **Alta**: Una vez completado el formulario de alta se valida que no exista el cliente. Si no existe, se guardan los datos en los archivos correspondientes.
- **Baja**: Modificar el campo eliminado en la estructura y guardarlo.
- **Modificación**: En la modificación se ingresa el nro de cliente. 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 camp a modificar.
- **Listados**: Para los listados se abre el archivo y se carga en un array y se ordena por nombre de cliente antes de mostrarlo.
- **Consulta**: Para la consulta la metodología sería similar a listados.

Sub-Menu Administración de Productos:

- **Alta**: Una vez completado el formulario de alta, persiste el producto en el archivo.
- **Baja**: Modificar el campo eliminado en la estructura y guardarlo.
- **Modificación**: En la modificación se ingresa el nro de producto. 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**: Para los listados se abre el archivo y se carga en una lista.
- **Consulta**: Para la consulta la metodología sería similar a listados.

2- Ingreso Con User y Pass para clientes:

Esta pantalla pide que se ingrese Usuario y Contraseña del cliente, si el cliente existe comprueba que la contraseña sea correcta y muestra los datos del cliente. Si el cliente no existe << muestra mensaje >> y si el cliente existe pero la contraseña no es correcta << muestra mensaje >>

Sub-Menú de login exitoso

- **Ver perfil:** Muestra la información completa del cliente logueado.
- **Mostrar pedido:** Muestra el listado de productos pedidos por el cliente con toda su información.

- **Agregar Producto:** Se ingresa el id del producto y se agrega en la lista del pedido del cliente.
- **Productos recomendados:** Muestra un listado de los productos recomendados para el cliente en base a su lista de pedidos. (Queda a criterio del equipo el algoritmo de recomendación).

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

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*				