Menú:

cout << "1 - “Levantar” los clientes del archivo “Clientes.bin”." << endl; cout << "2 -

Cargar Nuevo Usuario." << endl; cout << "3 - Buscar un usuario por ID." << endl; cout

<< "4 - Buscar un usuario por mail." << endl; cout << "5 - Desactivar un usuario

existente." << endl; cout << "6 - Listar todos los clientes activos ordenados por total

del importe." << endl; cout << "7 - Procesar un lote de compras." << endl;

cout << "8 - Mostrar por pantalla todas las compras realizadas de un cliente dado (desde el archivo procesados.bin)." << endl;

cout << "9 - Mostrar todas las compras realizadas entre dos fechas en un reporte escrito en

formato HTML. También mostrar el total de las compras. " << endl; cout << "10 - Mostrar

el mismo reporte del punto 9 en formato CSV." << endl; cout << "11 - Finalizar jornada." <<

endl;

Hipótesis planteada:

1. Creamos una función la cual abra el archivo en el cual íbamos a cargar los clientes y a su vez en esa, llamamos a otra la cual tiene 3 usuarios prediseñados los cuales se cargan en ese archivo abierto.
2. La función abre el archivo de los clientes y llama a otra función la cual pide al usuario ingresar los datos del nuevo cliente.
3. Es una función de tipo *bool* para que nos devuelva si lo encuentra o no. Esto lo hace abriendo el archivo y leyendo los *structs* en los cuales compara el id ingresado por el usuario con el del struct leído.
4. Hacer lo mismo que la función de buscar por *Id,* pero con el mail.
5. Busca el cliente por id y le cambia la variable que determina si el usuario esta activo o no, la cual es un *bool* que se cambiaria a *“false”.*
6. Abre el archivo y carga todos los clientes en un vector. Llama a otra función la cual se encarga de ordenarlos y al terminar esa, llama a otra la cual los muestra.
7. Una función bool la cual abre el archivo y llama a otra que contiene las solicitudes de datos para que el usuario ingrese.
8. Solicitamos el id del usuario el cual queremos mostrar sus compras, abrimos el archivo de compras las leemos y comparamos el id del usuario del struct de compra con el id ingresado anteriormente por el usuario, hasta que se termine el archivo. En caso de que la comparación se dé, llamamos a una función la cual muestra por pantalla la compra que fue comparada con éxito.

9. Solicitamos al usuario las 2 fechas entre las cuales haremos el filtro de los structs de compra. En la función creamos un archivo el HTML y abrimos el de procesados.bin. Imprimimos en el archivo HTML lo que serían los títulos y la tabla, todo con sus respectivas etiquetas, y hacemos un condicional el cual nos incluya todos los struct que verifiquen con las condiciones de este, para luego imprimir los structs en el HTML.
10. Hacemos lo mismo que en el punto anterior, pero en vez de abrir un archivo HTML lo hacemos con un CSV por lo cual, las etiquetas no estarían, solo quedarían los parámetros del *fprintf*. Estos parámetros determinan que tipo de variables se van a leer a continuación.
11. Finalizamos la jornada guardando los cambios en los archivos y cerrando el programa.

División de tareas:

Las funciones las fuimos repartiendo y en caso de estar trabados en alguna parte, las debbugeabamos o las pensábamos devuelta juntos.