Trabajo Práctico 2

Definiciones

Para este trabajo implementaremos una **aplicación de Windows Form** con persistencia en Archivos.

Detalle de la interfaz

Al ingresar el usuario ve una pantalla de inicio de sesión donde se pide usuario (DNI) y contraseña. Además se cuenta con opciones para Iniciar sesión, registrarse o configurar:

- Configurar:

Muestra una pantalla para seleccionar la carpeta donde se almacenarán los archivos. Por defecto se muestra la carpeta %user_profile%\ AppData\Local\Mercado.

- Registrarse:

Muestra los campos necesarios para agregar un nuevo usuario. Se debe validar que sean completados correctamente con manejo de excepciones. Una vez completado, al presionar para confirmar el registro, el usuario visualiza un cartel que indica si el registro fue exitoso.

- Iniciar sesión:

Ejecuta el método IniciarSesion de Mercado y luego consulta si esAdmin (ver en el siguiente punto):

En caso de ser administrador, ve una pantalla que permite visualizar todos los recursos del sistema en solapas y realizar ABM de los mismos según corresponda (por ejemplo las Compras no se pueden modificar los productos, solo los totales o eliminarse pero un administrador no podrá agregar una compra).

En caso de ser usuario, ve una pantalla donde:

- En el centro hay una tabla con todos los productos del mercado su precio individual. No se deben mostrar productos que no cuentan con stock.
- En la parte superior hay un cuadro para buscar productos por nombre y un switch para ordenar por nombre o precio (ascendente o descendente).
    - Al hacer doble clic sobre un producto se abre una nueva vista con el detalle del producto, un campo para seleccionar la cantidad deseada y un botón para agregar al carro. Al hacerlo el usuario ve un cartel indicando si se pudo agregar o no el producto.
- Sobre la columna izquierda se observan todas las categorías ordenadas alfabéticamente, hacer clic en una de ellas, el programa muestra en la tabla central los productos de esa categoría solamente.

- En la parte superior derecha hay un botón “Carro” que a su lado tiene una etiqueta que muestra la cantidad de productos que tiene agregados el usuario.
    - Al hacer clic en el carro, el usuario va a una nueva vista que muestra todos los productos del carro, el total de la compra y el botón para comprar. Se informa al usuario a través de un mensaje si la compra fue exitosa o no.

Condiciones para aprobar:

- La clase Form (interfaz gráfica/vista), debe tener un objeto de tipo Mercado con el que interactúa intercambiando datos. Los oyentes de esta clase se limitan a tomar datos introducidos por el usuario o a mostrar el contenido que devuelve Mercado pero **no implementan lógica alguna de procesamiento**.
- Se deben terminar de implementar los métodos para completar el circuito de compras (agregar al carro, comprar, etc).
- La clase Usuario ya no es abstracta ni se distingue entre Cliente Final y Empresa, ahora cliente agrega un atributo CUIT/CUIL directamente y otro atributo que indica si el usuario es Cliente, Empresa o Administrador.
- La clase Mercado agrega un método IniciarSesion que recibe como parámetro un DNI y clave, verifica si son correctos, en caso afirmativo retorna un entero “Id” que representa el ID del usuario (este dato debe almacenarlo Form para futuras consultas), caso contrario retorna -1. También agrega un método esAdmin que recibe como parámetro un ID de usuario y retorna verdadero si el usuario es administrador o falso caso contrario.
- Mercado debe agregar la persistencia de los datos en archivos. Al ser creado, el mercado tiene dos opciones:
    - Con consutructor vacío, busca, lee y crea los archivos en el directorio por defecto %user_profile%\ AppData\Local\Mercado (si la carpeta no existe, la crea, caso contrario lee el contenido).
    - Con constructor que toma como parámetro un string que indica dónde se deben leer/almacenar los archivos.
- Se debe crear un archivo por cada clase que interviene en el modelo (Usuario, Compra, Carro, Producto, Categoría). Dentro, se deben almacenar todos los elementos que existen en memoria, manteniendo la coherencia de los datos, es decir, las modificaciones realizadas en la aplicación deben impactar también en los archivos. La estructura interna de los archivos es a criterio del programador, podrían, por ejemplo, colocar un elemento por línea con atributos separados por un delimitador o bien que cada línea sea un atributo, etc.
- El programa debe contemplar todos los casos de excepción que se puedan producir, realizando un correcto manejo de excepciones. Por ejemplo, el archivo no existe, no se puede acceder para escritura/lectura, etc.
- Se deben liberar todos los recursos cuando finaliza la ejecución del programa.
- Implementar la clase Form cumpliendo con la descripción detallada en el punto anterior.

Condiciones de entrega

- La solución completa con todo el código se debe entregar en un archivo .zip con nombre TP2 – Grupo X. Reemplazar X por identificador del grupo (nombre o número).
- Se debe incluir dentro del .zip un archivo “ReadMe” que explique cómo utilizar ambos programas y cualquier aclaración que consideren necesaria o decisión de diseño que hayan tomado (puede ser en formato Word, txt, Excel, etc. No se evalúa presentación pero si contenido del mismo).
- La entrega se debe realizar por mail a mi casilla: walter.gomez@davinci.edu.ar
- El sujeto debe ser: Plataformas de programación: TP2 – Entrega grupo X. Reemplazar X por identificador del grupo (nombre o número).
- El cuerpo debe contener el detalle de los alumnos que componen el grupo, uno solo de ellos hace la entrega por el grupo pero debe poner en copia al resto, siendo responsabilidad de todos la entrega del trabajo.
- La fecha de entrega es 03/10/21. Entregar fuera de término afecta sobre la nota del trabajo práctico.
- Tengan en cuenta que la entrega se puede complicar ya que el mail no acepta archivos .EXE aun si los mismos están dentro de otro archivo .zip. También pueden compartir su proyecto por drive o enviarme el link de un repositorio.
- El proyecto debe compilar tal como fue entregado, caso contrario, será desaprobado.