Gabriel Remon
20 de junio de 2022

Sistema de atención al cliente de ferretería

Trabajo practico Nº4

Al abrir el programa este pedirá un usuario y contraseña, Dependiendo del tipo de usuario que ingrese, se contara con más o menos funciones dentro del mismo.

Funciones del programa:

1)Cargar nuevo pedido
2)Historial de ventas realizadas
3)Cargar nuevo cliente
4)Cargar nuevo empleado
5)cargar nuevo producto
6)Actualizar stock
7)Capacidad de generar una venta de un pedido pendiente
9)Mostrar lista
10)Mostrar lista de empleados
11)Todas las funciones del menuStrip Archivos
12)Todas las funciones del menuStrip Server

Permisos de cada usuario:

-Admin: acceso total de todas las funciones. Solo se podrá acceder al modo usuario antes de presionar alguno de los RadioButton. usuario: ”admin” contraseña: “admin”

-Empleado: funciones 1,2,3,7,11, El usuario será el DNI del empleado.
Usuario cargado “22333444” contraseña cargada: “123”

-Cliente: solo cuenta con las funciones 1 y 2 las cuales estaban vinculadas con su cuenta y su usuario será el DNI. Usuario Cargado “100204015” contraseña cargada: “123”.

Al salir del programa se guardara automáticamente todos los datos dentro de un archivo xml que estará en el mismo directorio que el ejecutable del programa y también se guarda toda la información dentro de la base de datos Ferreteria.

Al momento de iniciar la aplicación también se creara un directorio en el mismo directorio del ejecutable el programa llamado “Pedidos” donde se guardara por defecto los pedidos de productos que se realizaran a la distribuidora como un archivo txt con la fecha del día que se realizo ese pedido. Dentro del menuStrip se podrá definir una nueva dirección de guardo de este archivo.

Para poder realizar una venta primero debe realizarse un pedido, luego ese pedido estará pendiente para que el admin o un empleado lo cumpla (realizando doble click sobre el ítem de la lista de pedidos y pulsando la tecla "si"). Al momento de realizar el pedido la lista se actualizara y dejara de mostrar el pedido pendiente. Al mismo tiempo se agrega el pedido a la lista de ventas. Al momento de la realización del pedido los productos en stock quedan reservados para ese pedido por lo tanto se eliminan del stock total de la ferretería.

Temas aplicados en este trabajo practico:

Excepciones

Fundamentalmente se utilizo para mostrar los mensajes de error que pueden llegar a suceder en el programa. La mayoría fueron re lanzados por un bloque "try catch" modificando el mensaje de la excepción para que sea más entendible para el usuario el tipo de error. Ejemplo de su implementación.

-Linea 182 clase Biblioteca.Ferreteria (Relanzamiento de excepción con un mensaje)

-Linea 54 clase Forms.CargarPedido (Se muestra una excepción por un messageBox)

Pruebas Unitarias:

Se comprobó el funcionamiento de la mayoría de los métodos de ferretería ya que son los métodos mas complejos del programa y su buen funcionamiento significa que la mayoría de la funcionalidad es correcta. Ejemplo de su implementación:

-Todo el proyecto "TestFerreteria" (Por regla de estilo se realizo un proyecto por cada prueba de una clase)

Tipo genérico:

Se utilizaron los tipos genéricos para poder manejar los temas de archivos y serializacion, realizando una clase de tipo genérico para que dependiendo como del tipo de dato que se quiera guardar se cargue de una y otra manera. Reduciendo el codigo, La clase que implementa este tema es Archivos.IOArchivos. ejemplo de su implementación:

-Linea 132 clase Froms.Login

Interfaces:

Las interfaces de aplicaron en la clase Biblioteca.Persona con el fin de poder hacer las propiedades más genéricas de una persona.La interfaces se declaro en Biblioteca.Interfaces.Ipersona. Ejemplo de su implementación:

-Linea 10 clase Biblioteca.persona

Archivos:

Se utilizó la características d los archivos para definir el path predeterminados donde se guardara la configuración de la ferretería y el path de pedidos distribuidora, también se uso en la clase Archivos.IOArchivos para crear el txt de pedidos distribuidora. Ejemplo de su implementación:

-Linea 122 clase Forms.Inicio

-Linea 30 clase Archivos.IOArchivos

Serializacion:

Todo el trabajo se centro en guardar la instancia ferretería en un archivo xml para poder cargarlo posteriormente y recuperar los datos de la sección. Para poder realizar esto y no comprometer el encapsulamiento de las clases se uso el manejo de datos "Data Access Objet" y "Data Transfer Object", Las clases encargadas de este proceso fueron Archivos.IOArchivos, Biblioteca.DAO y Biblioteca.DTO.Ejemplo de su implementación:

-Linea 58 clase Archivos.IOArchivos

-Linea 178 clase Forms.Inicio

SQL y confección a base de datos:

En el mismo directorio donde se encuentra este pdf estar el script con toda la información de la base de datos de este proyecto, con las tablas y datos cargados. La clase Biblioteca.IO.Server es la encargada de realizar todas las conexiones a esta base de datos, enviando y recibiendo los distintos objetos de esta aplicación. Ejemplo de su implementación:

-Linea 209 clase Forms.Inicio

-Linea 151 clase Forms.Inicio

** Aclaración, Todo la aplicación pudo tener menos codigo y mejor funcionalidad si se hubiera echo mas énfasis en manejar los objetos por base de datos envés de archivos xml

expresiones lambda:

Se usaron mayormente en la implementación de hilos de ejecución dentro del programa, para reducir la cantidad de delegados que se devana utilizar. También se usaron las típicas expresiones "+=".Ejemplo de su implementación:

-Linea 75 clase Forms.PanelUsuarios

-Linea 105 clase Froms.Login

Delegados y eventos:

Estos dos temas se usaron en conjunto par poder hacer mas ágil la carga de los archivos xml a la aplicacion. También se utilizaron los eventos para enlazar los distintos eventos de los formularios así al momento de activar el evento de uno también realizaba una acción en otro.

Ejemplo de su implementación:

-Linea 29 clase Forms.Inicio

-Linea 75 clase Forms.Inicio

-Linea 183 clase Forms.Inicio

Programación Multi-hilo:

Se realizo dos hilos de ejecución en todo el programa los cuales deven actualizar cada un determinado tiempo los atributos de la aplicacion. Esto se pensó con la finalidad de que un cliente pueda estar conectado al mismo tiempo que el admin y que al momento de que el cliente cargue un pedido el admin lo vea, para esto se debe recargar la base de datos cada un cierto periodo de tiempo y comprobar los datos. Esto tendrá mas relevancia en futuras actualizaciones cuando se implemente mas las bases de datos. Ejemplo de su implementación:

-Linea 75 clase Forms.panelUsuarios

-Linea 105 clase Forms.Login

Métodos de extensión:

Se utilizo el método de extensión para agregar una codificación "SHA256" a los string de esta forma se podrá guardar con mas seguridad las contraseñas dentro de la base de datos. Ejemplo de su implementación:

-Linea 248 clase Biblioteca.IO.Server