

Aplicación de Atención al Cliente para Venta de Vuelos

Nuñez_IgnacioNatanael_2C_TPFinal

Esta documentación proporciona información detallada sobre cómo utilizar la aplicación y sus características clave.

Introducción: El uso de esta app está orientado a la venta de viajes específicos a clientes/pasajeros que deseen obtener un vuelo. Para que puedan comprar uno primero es necesario que se encuentren cargados en el sistema sus datos.

Requisitos del Sistema

Instalación de la Aplicación













Para utilizar este proyecto de visual studio debe de tener en el directorio @\Nuñez_IgnacioNatanael_2C_TPFinal\bin\Debug\net6.0-windows\datos

Los archivos: PASAJEROS_DATA.xml y Usuarios.json (ya vienen incluidos)

También es necesario levantar una base de SQL llamada:
IgnacioNatanael_2C_TPFinal , la cual contiene la tabla ViajesVendidos

(Dejo .bak y MDF)

Carpeta contenedora de archivos: *Archivos backup necesarios y base de datos*

<input type="checkbox"/> Nombre	Fecha de modificación	Tipo	Tamaño
 .git	23/11/2023 15:12	Carpeta de archivos	
 .vs	19/11/2023 22:34	Carpeta de archivos	
<input checked="" type="checkbox"/>  Archivos backup necesarios y base de ...	23/11/2023 15:18	Carpeta de archivos	
 Entidades	22/11/2023 16:30	Carpeta de archivos	
 JSON	22/11/2023 16:58	Carpeta de archivos	
 Nuñez_IgnacioNatanael_2C_TPFinal	23/11/2023 14:37	Carpeta de archivos	
 TEST_UNITARIOS	23/11/2023 15:10	Carpeta de archivos	
 TestResults	23/11/2023 15:11	Carpeta de archivos	
 .gitattributes	21/11/2023 17:44	Archivo de origen ...	3 KB
 .gitignore	21/11/2023 17:44	Documento de te...	7 KB
 Nuñez_IgnacioNatanael_2C_TPFinal.sln	23/11/2023 14:53	Visual Studio Solu...	3 KB
 README.md	21/11/2023 20:46	Archivo de origen ...	1 KB

Contenido

La app contiene varios forms en donde se pueden realizar diversas operaciones:

Form_Login: Este formulario se encarga de utilizar el archivo Usuarios.json para corroborar que el colocado sea correcto. Verifica su perfil e ingresa, Mismo cuenta con tres botones de debug (para tester) agilizando la entrada.

Form_Inicio: Este formulario es el principal y se encarga de mostrarte una interfaz con botones para el ingreso a diversos formularios.

En la esquina superior derecha se muestra el nombre del usuario, si este es clickeado se abrirá otra instancia de la app (Esto es por si requieren utilizar otro tipo de usuario al mismo tiempo)

Form_EliminarPasajeros: Este formulario se abre con el botón (Pasajeros) de la lista de modificadores, solamente se utiliza para eliminar pasajeros o visualizar los mismos. Esto es para arreglar problemas operativos (humanos) que haya tenido el vendedor.

Form_VentaViaje: Este formulario se abre con el botón (Vender) del form principal. Cuenta con todo lo necesario para rellenar datos y especificar el vuelo deseado hasta finalizar la compra. Mismo cuenta con un botón para buscar el pasajero comprador, en caso de no encontrarlo se abrirá el formulario

Form_CargaPasajero, Éste puede ser abierto de esa manera o apretando en el botón de generar Pasajero(cliente). Una vez generado o buscado el pasajero de manera correcta ya puede realizarse la venta

Form_ViajesInfo: En este formulario se ubican todas las ventas realizadas, son traídas desde el SQL

Implementación de Temas Específicos

Manejo de Excepciones: Además del manejo de excepciones en varias áreas del programa, el mismo contiene una personalizada con el nombre `SaltoSeguridadInputs`, Ésta es específica para dar a entender que el usuario fue capaz de saltar validaciones en campos de ingreso de datos y se lanza en el último botón de aceptar.

Ejemplo1: Form_CargaPasajero - btnAceptar_Click

Ejemplo2: Form_VentaViaje - ButtonAceptar_Click, en este tiene habilitada la opción de poder cambiar el nombre y apellido del pasajero. Esto es exclusivamente a propósito para permitir dejarlo vacío y provocar que lance la excepción a modo de prueba.

Extra: Las excepciones fueron manejadas en las capas críticas.

Gestión de Datos:

Utilización de archivos XML para almacenar y cargar datos de pasajeros y reservas.

Manejo de Genéricos: Esto se puede ver reflejado en el manejo de archivos como el Deserializar de JSON (*ManagerFileJSON.cs*) usado para usuarios.js dentro del formulario Login. El uso de XML fue hecho exclusivo para pasajeros sin genéricos

Manejo de Eventos y delegados en conjunto con Hilos: Se implementó el delegado *MensajeMostradoEventHandler* en el *GestorMensaje.cs*, este define la firma del método para el evento *MensajeMostrado*. lo que convierte el evento en una instancia de ese delegado. Se utiliza el evento lista de esos métodos. Al invocar el evento, se ejecutan todos los métodos en esa lista.

El Invoke asegura que si no hay ningún suscriptor, no se producirá un error de referencia nula.

Esto con el fin de poder utilizar el formulario **Form_Mensaje** que se muestra en un hilo secundario de interfaz de usuario. Con el fin de simular el guardado de los archivos en manera separada al uso del programa.

En resumen, *MensajeMostrado* es un evento y *MensajeMostradoEventHandler* es el tipo de delegado asociado a ese evento.

Manejo de Extensiones: Se implementó el evento *StringExtensiones.cs*, Es utilizado de manera exclusiva para verificar que el nombre y apellido del pasajero al momento de ser cargado contenga únicamente letras, evitando números. Puede verse implementado en el archivo *Validaciones.cs*.

SQL: Está implementando como base de almacenamiento de los viajes vendidos, en el **Form_ViajesInfo** se realiza una consulta para traer esas mismas ventas guardadas en la base. *BaseDatosSQL.cs*

Expresiones lambda: En el formulario **Form_VentaViaje** se encuentra cerca de la línea 325 Metodo *BuscarPasajeroPorDni()*, se utiliza expresion lambda para encontrarlo dentro de la lista luego de deserializar.

Test Unitarios: El proyecto cuenta con test para dos funcionalidades class *CalculadoraPreciosTests* y class *ValidacionesTests*.

La primera valida que la operación aritmética que utilizó para calcular el valor de los viajes se realice de manera correcta. Y la segunda prueba mis funciones que utilizo para validar campos de entradas (Al momento de cargar un pasajero principalmente)

Extra:

- En el login.cs se utilizó equals para verificar el correo y clave proporcionada con la lista de usuarios recorrida por un form
- La clase Pasajero hereda de Usuario
- Al momento de "acceder" al atributo de kilometraje de los enums destinos en *Form_VentaViaje* método *InicializarComboBoxDestinos()* línea 70, se utilizó la expresión ? operador de propagación de nulos (null-conditional operator) y ?? operador de fusión nula (null-coalescing operator).