Fernando Dieguez
TP FINAL
Sistema de registro y análisis de estadísticas para incidencias

Se buscó generar una solución que permita el registro de **incidencias** dados **usuarios** y **errores** existentes y hacer un análisis de las mismas, consultar, guardar o eliminar en la base de datos los usuarios y exportar las estadísticas como json.

El programa permite exportar los análisis de porcentajes de incidencias según algunos criterios aleatorios.
Los análisis pueden ser realizados utilizando cualquier tipo de predicado actuando sobre el listado existente de incidencias.

Se adjunta un proyecto database con un script de carga inicial y la tabla de usuarios
Se adjunta un archivo BACPAC con la base y la tabla creadas

Temas aplicados:
Excepciones: Se utilizan excepciones propias en distintas partes de la aplicación para mostrar mensajes específicos para errores específicos. Se lanzan excepciones en los casos necesarios y se atrapan mediante bloques try catch finally.
Dónde: Algunos ejemplos son: NucleoDelSistema.cs, Usuario.cs, Db.cs, Error.cs, etc

Pruebas Unitarias: Pruebas realizadas sobre clases clave del sistema, sobre extensiones que serán las encargadas de realizar el análisis según los predicados dados.
Dónde: Proyecto MSTestProyect

Tipos genericos: Se utilizan en toda la aplicacion tanto los tipos genericos de la biblioteca System.Collections.Generic como tipos genericos personalizados a fin de utilizar una misma lógica para distintos tipos de datos.
Dónde: ErrorDetail.cs, Extensiones.cs, carpeta Interfaces, etc.

Interfaces: Se utilizan interfaces que obligan a la implementación de métodos en ciertos casos, tales como clases que podrán ser exportadas, como clases que serán registradas en el sistema por medio de métodos específicos.
También se utilizan como genéricos, para poder realizar la misma lógica en algunos métodos con clases que implementen dichas interfaces.
Dónde: Carpeta interfaces, Extensiones.cs

Archivos: Junto con interfaces se realizaron clases que tienen permitido su guardado en forma de archivo, serializando las mismas como xml o json
Dónde: IExportable.cs, Usuario.cs

SQL: Conexión a base de datos local para el guardado , consulta y eliminación de usuarios de una base de datos.
Dónde: DB.cs

Delegados: Se utilizaron delegados para los callbacks de métodos que requieren ser ejecutados en el hilo que generó el control, para la carga de eventos o para ser enviados

como parámetros para métodos de filtrado.
Dónde: Extenciones.cs, MainForm.cs

Eventos: Suscripción de métodos a eventos ya existentes de winforms, generación de eventos para escuchar por cambios de datos y recargar los controles que muestran estos datos.
Dónde: MainForm.cs

Hilos: Uso de hilos para generar un paralelismo o concurrencia, para no bloquear la UI y utilizar de mejor forma los recursos asignados al proceso.
Dónde: MainForm.cs

Métodos de extensión: Se crearon métodos de extensión para filtrar cualquier tipo de colección, similar al comportamiento del Where de LINQ, otros métodos para obtener el porcentaje dados dos enteros y métodos que validan datos para los Enums.
Dónde: Extensiones.cs

La aplicación tiene un error actualmente, al utilizar los formularios para realizar las acciones se envía el formulario principal hacia atrás de las pantallas que haya activas en la PC que se encuentra ejecutando el programa.

Errores conocidos:
Para solucionar esto, se generó el main form con patrón singleton, para obtener una referencia al form principal y así poder llamar a su método ShowMainForm() o suscribir este método a un evento que escuche por estos cierres, pero haciendo debug del método funciona correctamente. En una ejecución normal sin breakpoints esto no es así.
Por lo tanto infiero que se trata de un mal manejo de los hilos, no se está esperando antes de ejecutar la instrucción y el formulario está perdiendo el foco.