Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Programación Genérica y Eventos: Informe – Tarea 8 – Parte 4**

* **Asignatura:** Programación Genérica y Eventos.
* **Profesora:** Mónica Nano.
* **Alumnos:** Tomas Molina y Edgar Karpowicz.
* **Tema:** Tarea 8 – Informe – Parte 4.
* **Fecha:** 20/10/24

**Informe: Tarea 8 – Parte 4**

* **Integración de Gestión de Recursos y Manejo de Excepciones:**
* **Introducción:**

En el Presente Informe, para la Materia de Programación Genérica y Eventos de la Carrera Ingeniería Informática en la Universidad Blas Pascal, se realizará una integración de todos los conceptos de Gestión de Recursos y Manejo de Excepciones en el Lenguaje de Programación C#, todos vistos y desarrollados en los anteriores escritos, para así dar fin a la Tarea 8.

Este Informe en particular, el cual tiene adjuntado un Programa que integra todas las funcionalidades desarrolladas en las partes anteriores en una sola aplicación de consola robusta, se realizara una investigación a profundidad sobre un tema avanzado en particular, como se realiza la Implementación de un Sistema de Logging Avanzado en Aplicaciones C#. Para posteriormente, enunciar como se implemento un Sistema de este estilo en el Programa Integrador.

* **Sistema de Logging Avanzado en Aplicaciones C#:**

En las Aplicaciones de C#, realizar algún tipo de Logging es fundamental, dado que permite realizar “debugging” de problemas, monitorear la salud de nuestros programas, y entender el comportamiento del usuario. Un Logging efectivo puede ahorrar mucho tiempo en las fases de desarrollo y ayudar a identificar rápidamente a problemas que se puedan estar dando.

Por ende, debido a la gran importancia del Logging en C#, a lo largo del tiempo han surgido diferentes formas de realizar el mismo, desde Patrones a diferentes Librerías que permiten cumplir con este importante propósito. Entre estos podemos mencionar:

* **Patrones:** uno de los Patrones más básicos que existen para realizar Logging en los Programas de C#, es el de Singleton. Este asegura que solo una Instancia del Logger sea creada y utilizada por toda la Aplicación. Por ende, es particularmente útil para casos en los que se posean grandes aplicaciones con múltiples clases o componentes que necesitan loggear información al mismo archivo.

Este tipo de Patrón fue el utilizado para implementar el Sistema de Logging en el Programa Integrador. Para ver un ejemplo de este, diríjase a la Sección Final del Informe que trata sobre el Programa desarrollado.

* **Librerías:** son varias las Librerías que existen para implementar Sistemas de Logging en C# de manera rápida y eficiente sin tener que reinventar las funcionalidades que se vayan a utilizar. Algunas de las que se pueden mencionar son:
* **NLog:** plataforma Open-Source para realizar Logging en .NET. Diseñada para poder escribir o loggear a múltiples objetivos, tales como la Consola, Base de Datos, Archivos .txt, entre otros. Así mismo, posee una robusta configuración que permite interactuar o cambiar el comportamiento del Sistema durante el funcionamiento.

Posee un Sitio WEB el cual explica como descargarlo, instalarlo y configurarlo. Cabe mencionar que se encuentra disponible para: .NET 5, 6, 7, 8, Framework 3.5 al 4.8, Core del 1.0 al 3.1, Xamarin Android + iOS, UWP, Mono 4, entre otros.

Se describe a si misma como de alto rendimiento, fácil de utilizar, fácil de ampliar, y flexible de configurar.

**Ejemplos:**

[**https://github.com/NLog/NLog/tree/dev/examples**](https://github.com/NLog/NLog/tree/dev/examples)



**Ilustración 1 - Logo NLog**

* **Log4Net:** es otra Librería que actúa como una herramienta para que el programador pueda sacar statements de log a una variedad de objetivos. Esta diseñada para la velocidad y flexibilidad, y posee múltiples características las cuales van desde: configuración dinámica, diseño modular, arquitectura probada, configuración XML, entre otros.

Como NLog, posee un Sitio WEB en el cual podemos acceder a toda su documentación y a un sinfín de tutoriales. En el caso especifico de esta librería, esta soporta a: .NET Core 6 o más, y Microsoft .NET Framework 4.6.2 o más alto.

Así mismo, posee un repositorio de GITHUB con ejemplos de implementaciones de esta Librería.

**Ejemplos:**

[**https://github.com/apache/logging-log4net/tree/master/examples**](https://github.com/apache/logging-log4net/tree/master/examples)



**Ilustración 2 - Logo log4net**

* **Ventajas de Librerías de Logging:**

Con algunas de las múltiples Librerías de Logging que existen en el mundo del C# mencionadas, uno rápidamente se podría preguntar cual es la ventaja de estas frente a una implementación casera básica que simplemente vaya escribiendo en un Archivo o subiendo a una base de datos a medida que el programa se ejecuta. Las respuestas son múltiples, pero las más importantes que se podrían mencionar son:

* **No reinventar la Rueda** (Si ya existe un Framework y Librería que implementan las Funciones que uno necesita, porque crearlas desde cero si las ya creadas no solo vienen testeadas, pero también bien mantenidas).
* **Rendimiento** (Con la Librería se puede rápidamente identificar el cuello de botella de rendimiento. Ya que averiguar que el Código de uno está funcionando lento debido a que se acabó el espacio en disco, se está escribiendo a un Archivo sin buffering, comprimir Archivos, entre otros, puede resultar problemático).
* **Funcionalidades** (Normalmente los Framework de Logging ya vienen con Funciones que podrían ser más difíciles de implementar, tales como aquellas que permiten escribir a una base de datos, a diferentes archivos, a un syslog, entre otros).
* **Mayor Flexibilidad y Modularidad** (Las Librerías de Logging ya vienen preparadas para ser lo más flexibles y modulares con el Código con el que se supone que interactúen. Por ende, no importa lo que uno agregue a su Código y/o cambie, siempre van a seguir funcionando de manera correcta).
* **Entre otras razones.**
* **Buenas Prácticas para el Logging:**

Algunas de las buenas prácticas que se pueden tomar o tener en cuenta al momento de trabajar o desarrollar un Sistema de Logging en ya sea Aplicaciones de Consola y/o más complejas serían:

* **No reinventar la Rueda** (Si bien uno puede crear su propia Librería de Logging, utilizar las opciones más probadas y testeadas siempre es una mejor idea).
* **Escribir Logs a Archivos, no a la Consola** (Hacer Logging en la Consola, si bien es útil, no sirve para mantener un orden de los eventos, comprar resultados, o realizar debugging. Por lo tanto, es mejor siempre escribir a Archivos, y si uno puede hacerlo en diferentes lugares para que estén lo más organizados posibles, mejor).
* **Hacer uso de Niveles de Logging** (Varias Librerías, particularmente log4Net, contienen la habilidad de ignorar mensajes no importantes y de darles mayor prioridad a otros, en escenarios específicos al momento de logear, estas son: Warning, Error, Fatal, etc. Hacer uso de esta característica puede ahorrar mucho tiempo al momento de identificar errores o realizar troubleshooting).
* **Filtrar los Mensajes de Logging** (Otra característica que poseen los Framework de Logging como log4net es que se pueden filtrar los mensajes para acotar los mismos y recibirlos cuando uno lo requiere o contienen información importante. Algunos filtros que existen en log4net por ejemplo serían: log4net.Filter.DenyAllFilter, deniega a todos los mensajes, log4net.Filter.StringMatchFilter, nos envía los mensajes que tengan un String en particular, log4net.Filter.LevelRangeFilter, según los niveles de logging que se hayan puesto, entre otros).
* **Tener en cuenta los Problemas de Logging** (Uno de los mayores problemas que posee el Logging, es que tiene un alto consumo de rendimiento y puede llegar a exponer información sensible. Por lo cual, siempre hay que asegurarse que se este implementando al mismo cuando sea necesario y de una manera adecuada para mitigar lo más posible a estos problemas).
* **Programa Integrador:**

Acompañando a este Informe, se encuentra un Programa el cual integra las Actividades realizadas para los anteriores Escritos y Partes de esta Tarea en uno solo. Es decir, combina la Calculadora y FileManager realizados anteriormente.

La principal característica de este Programa Integrador es que implementa un Sistema de Logging Singleton, el cual permite guardar Datos, Excepciones, entre otros, en un Archivo .txt Local.

Este Sistema de Logging, fue creado y diseñado para este Ejercicio en particular sin hacer uso de un Framework / Librería en específico. Por lo que, si se implementase el mismo programa con una Librería tal como log4net, podría resultar en una mejor experiencia en tanto rendimiento como flexibilidad y funcionalidad, ya que se podrían configurar múltiples otros aspectos tales como cuales mensajes se deben recibir en cierto momento, cual es la prioridad, entre otros.

Se incluye Imágenes del Código implementado del Logging:

Captura de pantalla de computadora

Descripción generada automáticamente

**Ilustración 3 - Singleton realizado para Logging**

Captura de pantalla de un celular

Descripción generada automáticamente

**Ilustración 4 - Extracto de Código en el cual se hace uso del Singleton**

Así mismo, se encuentran la Calculadora y FileManager, las cuales se pueden acceder directamente desde un menú principal el cual nos permite acceder a ambos programas. Todos se encuentran implementados con Control de Excepciones y con el Sistema de Logging.

* **Recomendaciones:**

Como Recomendación principal, se podría sugerir el mayor uso de Librerías de Framework de Logging, y, por otro lado, se podría mencionar el hecho que el Logging es una Actividad bastante costosa en términos de rendimiento, y que si se implementa de manera inadecuada puede llevar a problemas. Así mismo, podría ser interesante ver como es el uso de una de estas Librerías de Logging y como interactuaría un Programa de C# con una Base de Datos en la Nube.

* **Bibliografía:**
* <https://learn.microsoft.com/en-us/dotnet/core/extensions/logging?tabs=command-line>
* <https://www.papertrail.com/solution/tips/7-best-practices-for-c-logging-with-examples/#:~:text=Logging%20in%20C%23%20applications%20is,identify%20and%20fix%20production%20issues>.
* <https://dotnettutorials.net/lesson/singleton-design-pattern-real-time-example/>
* <https://nlog-project.org/>
* <https://logging.apache.org/log4net/index.html>
* <https://github.com/apache/logging-log4net/tree/master/examples>
* <https://stackoverflow.com/questions/144682/what-is-the-point-of-using-a-logging-framework>