2 DE JULIO DE 2017

EJECUCIÓN DE COMANDOS Y GESTOR DE PROCESOS

INFORME DEL PROYECTO

MENA MAX
ABARCA VIQUEZ ANDRES

ORGANIZACIÓN DEL EQUIPO

Para este proyecto se optó por hacer uso del lenguaje orientado a objetos C#, usando el framework de Microsoft .NET, para así lograr una solución con un tiempo de respuesta alto y un fácil acceso. Debido a los requerimientos de desarrollo, decidimos unánimemente y de manera equitativa, que el visualizador, las estadísticas, y las actualizaciones a los procesos del sistema operativo dentro de la aplicación, serian desarrollados por Andrés; y el módulo visualizador de los servicios, la actualización de los estados de los servicios y respectiva documentación, serian desarrollados por Max.

La solución se trabajó en la plataforma de desarrollo de Visual Studio de manera individual, pero en la misma solución, haciendo uso de los servicios de repositorio de github, para la unificación de código e implementación en tiempo real de dos módulos diferentes a esta solución.

Planteamiento

La solución propuesta permite la interacción con los procesos y servicios que se ejecutan a nivel operativo de Windows, en el lenguaje de programación C# por medio de Windows Forms, haciendo uso de la librería .NET 1.1 de Microsoft y demás versiones en adelante. El proyecto plantea la necesidad de monitorear la actividad de los procesos como objetos, mediante la obtención de la información de la máquina por medio de la librería System.Diagnostics.Process de este framework y proporciona información descriptiva de sus atributos, como los son: nombre, identificador, cantidad porcentual de recursos utilizados, el nombre del usuario y su prioridad. Provee datos estadísticos de los procesos, así como, por ejemplo: el porcentaje de procesamiento utilizado únicamente en procesos, y el porcentaje de espacio de memoria promedio de todos los procesos en ejecución y la capacidad de memoria de la máquina.
Permite además cuatro operaciones diferentes sobre los procesos que la máquina está ejecutando, los cuales son: finalizar un proceso individual, finalizar el árbol de procesos, actualizar la prioridad de un proceso individual y actualizar la afinidad de un proceso individual.

Los servicios del sistema son igualmente manejados como objetos mediante acceso a los registros del sistema operativo, mediante la librería System.ServiceProcess de este framework, y asimismo proporcionan información descriptiva de los atributos, los cuales son: identificador, nombre, descripción, usuario, prioridad y estado.
Al igual que en la parte de los procesos del proyecto, este permite las tres siguientes operaciones sobre los servicios: iniciar, pausar o detener algún servicio, dependiendo del estado actual de cada servicio de manera individual.
Guía de desarrollo del proyecto

GUIA DE DESARROLLO DEL PROYECTO

- En la solución presente se implementaron las siguientes técnicas:
  Se inicia con la creación de los dos principales objetos de todo el proyecto, para los procesos “ProcesosClass” con los atributos solicitadas por el enunciado:
    - Identificador

- Nombre
- Descripción
- Recursos utilizados
  - CPU
  - Memoria
- Usuario
- Prioridad

- Igualmente, para los servicios “ServicioClass” con los siguientes atributos:
  - Identificador
  - Nombre
  - Descripción
  - Usuario
  - Estado

- La interfaz del proyecto se establece en un Windows Form principal de tipo MDIParent el cual provee una interfaz de múltiples documentos para el manejo eficaz y fácil del usuario, se crean los forms hijos, los cuales son las representaciones gráficas donde se va a mostrar la información manipulada tanto de los servicios como de los procesos. Se utiliza un formato estándar en ambas pantallas, y la información de los procesos y servicios se presentan por medio de un dataGridView dentro de un WindowsForm hijo, individual.

- Para resolver el problema de donde obtener los datos de los procesos del sistema operativo, se utilizan las librerías “System.Management, System.Management, System.Diagnostics” con esta se solicita la información de los procesos directamente de Windows. A continuación, se describen las funciones creadas para el uso y desarrollo correcto de la aplicación:

  - Función cargarProcesos()
    - Función que obtiene por medio de la clase Process de la librería Diagnostic, todos los procesos actuales ejecutándose del sistema operativo de Windows. Aplica para máquinas con sistemas operativos de Windows 7 en adelante. Las informaciones de los procesos son almacenadas en objetos tipo procesoClass y los

almacena en un dataTable. Después cuando todos los objetos son cargados a la data table, el dataGridView, despliega en el form la lista de objetos.

- Función getNombreUsuarioProceso(string procesoID)
  - Función que obtiene el nombre de usuario por medio del windows management instrumentation, el cual provee una infraestructura de fácil acceso a la información de la máquina y desempeño. Extrae el nombre y el dominio del usuario, y retorna solamente el nombre de usuario.
- Función getPorcentajeCPU(string nomProceso)
  - Función que obtiene el porcentaje del CPU utilizado, obtenido de la interfaz de programación "Performance Counters" que provee información de cómo y que tan bien se desempeña el sistema operativo, la red y otros dispositivos. Por medio de varios accesos a este contador, se sustrae el peso de procesamiento porcentual, dedicado a el proceso con el nombre recibido en el parámetro. Retorna un String del valor porcentual.
- Función getPorcentajeCPUTotal()
  - Función que calcula el porcentaje de CPU usado en todos los procesos. Utiliza la misma información y programación de la función anterior, sin embargo, evalúa al CPU respecto a todos los procesos.
- Función memoriaProceso(long memoria)
  - Calcula la memoria actual de un proceso. Recibe la cantidad de memoria de un solo proceso, y retorna un string de la memoria, en un formato entendible, como por ejemplo 1000 bytes = 1kb.
- Función memoriaTotal()
  - Calcula la memoria total ocupada por todos los procesos. Es la sumatoria de la memoria utilizada de todos los procesos, se divide en la cantidad de procesos habilitados y se hace un llamado a la función anterior para retornar un valor entendible de memoria utilizada.
- Función cerrarArbolProcesos(string nombreProceso)
  - Cierra el árbol de procesos y subprocesos. Este recibe el nombre del proceso padre a buscar. Carga en una lista de objetos de tipo Process, todos los procesos con ese mismo nombre, y de manera

individual los cierra haciendo uso de la clase padre “Process” haciendo uso de la función “Kill”.

- Función cambiarPrioridad(int prioridad)
  - Método para ajustar la prioridad del proceso. Este recibe por parámetro el identificador único del proceso y despliega un menú para seleccionar alguna de las opciones disponibles, las cuales son: baja, por debajo de lo normal, normal, por arriba de lo normal, alta, y en tiempo real. Dependiendo de la selección del usuario, se obtiene el objeto tipo Process, y se usa la función de la clase padre para actualizarlo llamado: “PriorityClass”
- Función traducirPrioridadProceso(string prioridad)
  - Función para traducir la prioridad de un proceso de Ingles a Español. Esta se utiliza ya que la clase padre “Process” provee la prioridad del proceso en Inglés. Para mayor facilidad al usuario esta información de prioridad se despliega en Español.

Para resolver el problema de cómo iniciar un proceso mediante la aplicación, se hace uso de una función de la clase padre llamada Process.Start.

- Dentro del form llamado: “Ejecutar proceso” se maneja el evento capturado del botón “Ejecutar” e instancia un objeto tipo Process mediante el nombre ingresado del proceso por medio del usuario, valida si este proceso se encuentra en ejecución. Si ya se encuentra en ejecución, le pregunta al usuario si quisiera volverlo a ejecutar, de lo contrario, ejecuta el proceso. Siempre y cuando el nombre del proceso ingresado sea válido.

- Para el manejo de la solicitud de los datos de servicios se utilizan estructuras muy similares a las utilizadas para obtener los datos y realizar los cálculos de los procesos, las cuales consultan principalmente la clase ServiceController que permite obtener la información directamente del sistema operativo utilizando además el soporte de las clases que se usaron para obtener los datos de los procesos “System.Management, System.Management, System.Diagnostics”.

- A continuación, se detallan una lista de los objetos implementados para la obtención de la información que se despliega en el DataGridView del WindowsForm hijo elaborado para los los datos y resultados de los cálculos de los Servicios;
    - Función cargarServicios()
        - Función que obtiene por medio de la clase ServiceController de la librería Diagnostic, todos los servicios actuales ejecutándose del sistema operativo de Windows. Aplica para máquinas con sistemas operativos de Windows 7 en adelante. Las informaciones de los servicios son almacenadas en objetos tipo servicioClass y los almacena en un dataTable. Después cuando todos los objetos son cargados a la data table, el dataGridView, despliega en el form la lista de objetos.
    - Función getPorcentajeCPUTotal()
        - Función que calcula el porcentaje de CPU usado en todos los servicios. la interfaz de programación "Performance Counters" que provee información de cómo y que tan bien se desempeña el sistema operativo, la red y otros dispositivos. Por medio de varios accesos a este contador, evalúa al CPU respecto a todos los servicios. Retorna un String del valor porcentual.
    - Función obtenerServicio(string nombreServicio)
        - Función que obtiene el ID del servicio por medio del nombre, se realiza una consulta LINQ a Win32_Service donde se obtiene como resultado el ID del proceso el nombre del usuario, se almacena en una instancia de la clase ServiciosClass.
    - Función iniciaListas()
        - Función que simplemente centraliza la instanciación de la lista con los datos de los servicios y el estado de los servicios de la lista ServiceController.

Resultados:

- Los resultados de este ejercicio resultan satisfactorios para el equipo, en la elaboración del proyecto se encuentra que existen distintas soluciones para obtener los datos requeridos para facilitar la información propia del sistema tanto en los servicios como en los procesos.
- Los cálculos para obtener los porcentajes de memoria y CPU son de dominio público con la diferencia de que los recursos del sistema no siempre lo son, además, por los permisos del sistema algunos recursos no son accesibles desde el FRAMEWORK de Visual Studio por lo que el resultado de estos cálculos varia en comparación con los datos que el Task Manager del sistema indica, pero estas diferencias no son relevantes para comprometer la funcionalidad del aplicativo.
- Con respecto a los servicios se implementas las opciones solicitadas en el enunciado del trabajo y se confirma su funcionalidad a nivel de depuración, sin embargo, la modificación en los servicios del sistema es una excepción, por lo que en la mayoría de las ejecuciones nos encontramos con errores por parte del sistema, debido a permisos de acceso.
- La sección de los procesos funciona de manera adecuada y la opción de modificar el estado de los procesos es más factible a nivel del sistema, por lo que se puede comprobar su funcionalidad en tiempo de ejecución sin buscar ejemplos específicos.

Conclusión:

Ambas partes consideramos que el trabajo presenta una gran oportunidad de aprendizaje y conocimiento de la posibilidad del consumo de datos directamente del sistema, pudiendo considerar este nuevo conocimiento como una herramienta valiosa a la hora de desarrollar nuevos aplicativos, que puedan requerir de alguna manera el evaluar el estado del sistema operativo desde el que se están ejecutando, para recomendar acciones que mejoren el desempeño o que nos permitan conocer limitaciones del aplicativo en mismo. Ambos consideramos que el aplicativo que se presenta cumple con todo lo requerido en el enunciado del proyecto, estamos agradecidos por esta oportunidad de aprendizaje y satisfechos con el trabajo realizado.