|  |  |
| --- | --- |
|  | **2013** |
|  | IES San Vicente  Jorge Bernabeu Mira |

|  |
| --- |
| **Total ComMAN-DAM-** |
| Documentación del proyecto final para el curso 2012-2013 de Desarrollo de aplicaciones multiplataforma |

**Resumen:**

# Total Comman-Dam- es un gestor de archivos el cual nos permitirá entre otras opciones copiar, eliminar y mover ficheros entre sus dos diferentes paneles. El programa está plenamente desarrollado en Visual Basic.NET ya que es uno de los lenguajes más cómodos de programar y su IDE, Microsoft Visual studio es uno de los mejores actualmente en el mercado, como bien ha demostrado al ganar los premios InfoWorld's 2013 Technology of the Year [1]. El proyecto anteriormente mencionado estará orientado a objetos y en una arquitectura de 3 capas, Presentación, Negocio y Datos.

**Memoria:**

**Introducción:**

Total Comman-Dam- es mi propuesta de final de ciclo que elegí al ver que bastante gente usaba el gestor de datos “Total Commander”, la idea nació de las ganas por crear algo que no habíamos visto demasiado en clase, como es manejar los ficheros, conseguir moverlos, copiarlos o eliminarlos. Después de ver que el mercado de gestores de archivos es importante, teniendo como referencia Total commander en Windows o Xplorer2 ambos están escritos en lenguajes de programación a bajo nivel, por lo tanto el resultado no va a ser el mismo. Mi programa va a estar en Visual Basic, un lenguaje de programación creado por Microsoft el cual está orientado a objetos, es sin duda el lenguaje perfecto por su facilidad, su simpleza y el maravilloso IDE que nos brinda Microsoft a los estudiantes para poder desarrollar programas en él.

Analizando las otras aplicaciones nos encontramos con que, Total commander, está desarrollado en Delphi1 y Delphi2. Su punto fuerte es que nos permite actuar con servidores FTP y acceder a particiones Linux desde nuestro disco en Windows. Por otra parte Xplorer2 está desarrollado en C++. Sus puntos fuertes no difieren mucho de Total commander, también nos permite conectar por FTP y dispone de una versión Lite o free para poder usarlo con normalidad.

La diferencia que tendrá mi programa con el resto, será la simpleza, ya que teniendo las opciones justas existen menos posibilidades de perderte entre tanta opción como en Total commander en el que el usuario medio no llega a usar todas las opciones. Añadir un poco de personalización es fundamental para un programa con un entorno visual tan cuidado como el de Windows por lo tanto será otro punto a favor.

**¿Por qué Visual Basic.Net?**

Visual basic ha sido uno de los lenguajes que hemos dado durante todo este año, que gracias a su simplicidad, a su facilidad para los programadores novel y que está creciendo y se sitúa entre los 15 lenguajes más usados en 2013 según la web comoprogramar.org[2] por eso he creído oportuno elegir este lenguaje que nos brinda Microsoft. Teniendo de referencia la magnífica página de ayuda de Microsoft el proyecto puede salir bastante bien.

**Objetivos:**

Visto que los más importantes gestores de archivos tienen unas cosas buenas y otras malas, mi objetivo es intentar juntarlos y crear una aplicación bastante más simple y con algunos toques de personalización, permitiendo cambiar el tamaño de la letra o los colores de nuestra aplicación y guardarlos según el usuario para que cada usuario tenga su propia personalización. Entre las opciones más interesantes de estos programas se encuentra copiar, mover, renombrar varios archivos al mismo tiempo, eliminar, comparar el contenido de dos carpetas y navegar entre ellas. Mi objetivo es crear una aplicación con las características “más” importantes de estos programas y las que más se usarían día a día. Entre las principales funcionalidades que tendrá mi programa se encuentran:

* Navegar entre los diferentes paneles
* Copiar contenido de un panel a otro
* Mover contenido de un panel a otro
* Renombrar varios archivos con el mismo nombre
* Crear nuevas carpetas y nuevos ficheros de texto
* Comparar el contenido de dos carpetas
* Comprimir y descomprimir carpetas
* Mandar archivos por correo
* Imprimir documentos
* Etc…

**Metodología, resultados y discusión:**

Para este proyecto utilice el modelo de 3 capas para darle mejor legibilidad al código y si en algún momento pretendo cambiar el entorno grafico que no afecte demasiado al funcionamiento de la aplicación. Por un lado he separado mi capa de diseño, metiendo en ella los formularios. En la capa de negocio tengo la clase encargada de distribuir los datos vengan desde la clase inferior que vengan y en la clase de datos tengo las clases encargadas de crear, leer, cambiar o generar datos.

Con la separación en clases del proyecto no encontré ningún problema, el formulario principal, tiene dos paneles, a los cuales hay que distribuir datos, en la capa de negocio. El formulario contendrá un objeto de la clase Total Commander que se encargara de distribuir los datos a los paneles izquierda y derecha.

TotalCommander

Form

Izquierda

Derecha

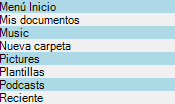
Al empezar el proyecto el primer problema que me encontré fue el de que controles usar en la parte visual para mostrar el listado de archivos, las opciones eran ListView, ListBox o TreeView. Al final me decante por ListBox, ya que sabía a grandes rasgos cómo funcionaba. Descarte TreeView porque no me gustaba el resultado final que le daba al programa. ListView me pareció extraño y el no saber si me iba a resultar útil, lo descarte.

El siguiente problema surgió al colocar los Listbox en el formulario y comprobar que estos no se redimensionaban al redimensionar el formulario, al final solucione el problema ajustando los elementos a los bordes y dejándoles un espacio. El método Resize del formulario principal me ayudo en esta tarea.

Para la mayoría de acciones de ficheros use los métodos de My.Computer.FileSystem que permiten con una simple llamada, copiar, mover, renombrar, borrar o determinar si el archivo existe. Esta clase contiene los mismos métodos tanto para ficheros como para directorios.

La mayoría del trabajo no tiene complicaciones, simplemente es costoso comprobar y validar todas las acciones posibles del usuario e intentar que no tenga fallos aparentes. Ya que el fichero debe estar en el momento correcto sin ocupar y comprobando que el usuario no lo haya borrado en el transcurso del proceso, distribuir los elementos oportunos etc… El error más difícil de comprobar fue que al acceder a ciertas carpetas de Windows nos saltaba una excepción diciendo que no tenemos permisos, después de probar a ejecutar el programa como administrador, darle permisos a las carpetas y bastantes cosas más que encontré por internet y que ninguna funcionó. Al final la opción que tomé como la más óptima fue, capturar la excepción y no dejar al usuario entrar a esas carpetas ya que ninguna de las opciones que encontré funcionó.

Tras varios días trabajando en ello conseguí que tuviera las principales opciones, como copiar, eliminar o mover. A medida que iba necesitando creaba los métodos oportunos tanto en la capa de datos como en la de negocio. Después de tener medio programa ya completado, descubrí que hubiese sido mejor usar un ListView, ya que permite organizar un poco mejor los datos y distinguir a simple vista cuales son carpetas y cuales son ficheros, ya que es el principal problema que encontré en el ListBox. Para medio solucionarlo, dibujé cada línea del ListBox “manualmente” y la pinto de un color diferente, para separar un poco las líneas y así ver mejor la diferencia.



En cuanto a los nombres repetidos en el sistema, decidí utilizar el mismo sistema que usa Windows y poner al final del nombre el número de archivo que le corresponde. Ej: nuevo.txt, nuevo(1).txt, nuevo(2).txt.

C:\Users\JorgeBern\Desktop\ProyectoFinal\nombres.png

Utilizo el mismo método para todas las acciones ya que me parece una buena idea, ya que es transparente al usuario, bastante tolerante y sirve para todas las funciones sin importar si copian, mueven o renombran.

Después de crear los métodos principales, decidí crear un hilo secundario para ejecutar las tareas más pesadas que podrían ser entre otras las de copiar, mover y comprimir y así que la aplicación principal no se congelara durante el proceso. El problema llegó al no saber realmente usar hilos o procesos en Visual Basic, buscando encontré varias soluciones, entre ellas implementar un elemento llamado backgroundworker, que permite ejecutar tareas en segundo plano, después de varias horas mirando ejemplos y probando, me encontré con el problema de que no podía acceder a un elemento grafico desde un hilo secundario, busque información acerca de los “Delegados” y como utilizar esos objetos desde otros hilos. Por fin, en un post Americano, encontré un ejemplo de lo que yo quería hacer, el problema es que era con hilos y no con BackgroundWorker. Al final tomé la decisión de cambiar todo el trabajo anterior con los delegados y el backgroundworker por esta solución más simple con hilos. La razón principal del cambio fue que con hilos y este comando funcionaba, el código quedaba más claro y no resultaría tan deficil de entender en un futuro. Las líneas de código clave son las siguientes:

Dim background As Thread

CheckForIllegalCrossThreadCalls = False

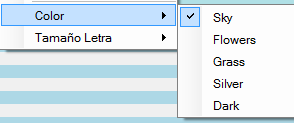
La primera crea el hilo en el que se ejecutaran las acciones usando el constructor background = New Thread(AddressOf Me.Copiar). Que nos permite usar la función especificada en segundo plano. La segunda línea nos permite que un hilo no sea invocado por varios objetos al mismo tiempo, solucionando el problema que teníamos anteriormente con el backgroundWorker.

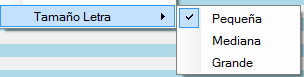
Al crear el método de Drag&Drop para copiar archivos de un panel a otro, me encontré con bastantes problemas, ya que bastante gente por internet decía que no se puede hacer drag&drop de varios objetos. Lo intente de mil formas y no hubo manera de conseguirlo así que el Drag&Drop es la única función que no acepta multiselección y que funciona un poco diferente. Por diferentes problemas no he conseguido que se ejecute en segundo plano aun.

A medida que avanzaba, encontraba otras cosas útiles que añadirle como, una pestaña de favoritos, rutas predeterminadas, ordenar los ficheros alfabéticamente o por fecha de creación, mostrar los archivos recientes etc…

Para la correcta asignación de archivos y que todo funcionase correctamente, pensé que debería de hacer un sistema para saber que panel ha sido el último en tener el foco ya que con varios controles no podemos obtener el foco del panel si estamos en un checkbox o cualquier control que nos permita tener el foco. Fue tan sencillo como crear una variable que cambiase cada vez que el panel recibía el foco poniendo a 0 para el panel izquierdo y 1 para el panel derecho permitiendo que por ejemplo, si seleccionamos el panel derecho y vamos al TextBox para filtrar, saber que el ultimo panel usado ha sido el derecho y que en este caso filtre bien el contenido.

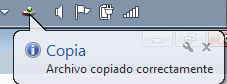
En el tema de la personalización decidí que lo mejor sería crear temas predeterminados y dejar al usuario que eligiera entre ellos. Varios pequeños cambios entre ellos, el color de la selección, el de fondo y las líneas para diferenciar los elementos. También implemente la elección del tamaño de la letra, para mejorar la integración con diferentes gentes con problemas de vista o simplemente para gente que le gusta el texto grande





Estas opciones se guardan para cada usuario, teniendo cada uno sus propias configuraciones y se cargan desde los ficheros de configuración cada vez que el usuario ejecuta el programa. Decidí que estas acciones no deberían de ir en la clase Panel ya que no son acciones específicas de los paneles por eso cree la clase Preferencias que se encarga principalmente de cargar las preferencias y los favoritos desde los ficheros de configuración.

Como opción complementaria a las acciones decidí añadir un Notify Icon para notificar que las acciones se habían completado ya que la idea de un Msgbox no me gusta, pues implica cortar el programa para que el usuario pulse el botón de aceptar.



Para guardar los favoritos decidí que sería bueno, que tuviesen un nombre de identificación, para una mejor identificación por parte del usuario. Estos favoritos también se guardan para cada usuario y son completamente personalizados. En la ventana de preferencias de nuestros favoritos, nos permite elegir cual será nuestra ruta predeterminada y en la cual empezaremos a navegar cada vez que iniciemos el programa. Para distinguir cual es la ruta predeterminada, esta se pinta de rojo en nuestros favoritos. Para administrar los favoritos tenemos las opciones de renombrado y borrado pues al añadir se podía convertir en una locura. Si un usuario borra su ruta predeterminada actual, esta directamente se convierte en la del escritorio. Las rutas principales como son Mis documentos, Escritorio o Archivos de programa no se podrán borrar ni modificar, son predeterminadas, es una solución para que el usuario no meta mucho la pata y siempre tenga claro por donde volver al escritorio.

Para listar los archivos y carpetas recientes que se usaron en el Pc he usado la ruta de acceso Environment.GetFolderPath(Environment.SpecialFolder.Recent) que obtiene la ruta entera donde se encuentran los archivos recientes y luego obtengo los archivos y os muestro en el listBox, me pareció una idea interesante ya que muchas veces cerramos un archivo y no sabemos dónde estaba. Estos archivos tienen las mismas funciones exactamente que los de un panel normal.

El siguiente método que me pareció bastante útil pero no estaba en las especificaciones principales es el de ordenar. Me pareció que era indispensable y gracias al método de la burbuja:

For i As Integer = 2 To archivos.Length

For j As Integer = 0 To archivos.Length - i

direInfo = My.Computer.FileSystem.GetDirectoryInfo(\_ruta & "\" & archivos(j))

direInfo2 = My.Computer.FileSystem.GetDirectoryInfo(\_ruta & "\" & archivos(j + 1))

If direInfo.CreationTime.CompareTo(direInfo2.CreationTime) >= 1 Then

aux = archivos(j).ToString

archivos(j) = archivos(j + 1)

archivos(j + 1) = aux

End If

Next

Next

Conseguí ordenar los archivos por fecha de creación. Y gracias a la propiedad Sort() de las listas conseguí en poco tiempo ordenar de varias formas diferentes. También implemente la opción de ordenar por tamaño, pero todas las funciones que hice y las que encontré, duraban demasiado. Ej: ordenar por tamaño una carpeta con 20 carpetas dentro, puede tardar de 30 seg a 1 min que es tiempo que el usuario esta sin hacer nada.

En la opción de comprimir usé la librería que nos proporciona Microsoft para comprimir carpetas en zip. Los métodos solo nos permiten comprimir carpetas, por lo tanto nos vemos un poco atados a la hora de comprimir. Este método no nos permite la multiSelección, con lo cual no estará permitida en esta acción. Para descomprimir usamos la misma librería que nos proporciona Microsoft.

Para comprimir usaremos:

ZipFile.CreateFromDirectory(original, destino, nivelDeCompresion, DirectorioIncluido)

Para descomprimir:

ZipFile.ExtractToDirectory(comprimido, descomprimido)

No utilicé librerías externas por el simple problema de que no sabía si las iba a poder exportar bien o si iban a funcionar en otros ordenadores. Utilizando las de Windows me aseguro que funcionen en todos los ordenadores con este sistema operativo.

Otra opción que me pareció bastante buena fue la de poder obtener la información de un archivo y mostrarla por pantalla, para ello me ayude de la clase FileInfo:

Dim info As FileInfo = My.Computer.FileSystem.GetFileInfo(fichero)

Obtenemos la información que luego podremos usar llamando al objeto y el tipo de información que queremos obtener. Ej: info.Extension

Para mostrar la información hice un formulario que parecido al de Windows. Después de esto, pensé que la información del Pc tampoco estaría mal, gracias a la clase Environment obtuve información del Pc como: nombre de usuario, nombre del Pc, numero de procesadores…

Para la creación de ficheros y carpetas, encontré un método que crea carpetas, pero no encontré ninguno para la creación de ficheros, así que decidí crearlo con un flujo de datos StreamWriter, creando un fichero vacío y cerrándolo al instante.

Probando cosas al final conseguí hacer que el Drag&Drop de varios elementos funcionara, funciona de una forma bastante peculiar, pero funciona. El problema era que estaba llamando al elemento recién pulsado y no a la colección de elementos que tenemos seleccionados. La peculiaridad es que debemos arrastrar justo al seleccionar el último elemento de los que queremos arrastrar.

Leyendo sobre los usos de Total Commander descubrí uno bastante interesante. Comparar las dos carpetas y mirar que carpetas faltan en otra carpeta. Para esto use la siguiente función:

For Each elemento As String In My.Computer.FileSystem.GetDirectories(\_ruta & "\")

existe = False

Dim partido As String() = elemento.Split(CChar("\"))

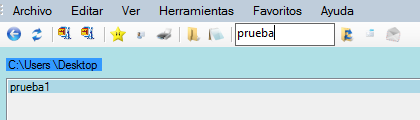
If Not My.Computer.FileSystem.DirectoryExists(destino & "\" & partido(partido.Length - 1)) Then

faltan.Add(partido(partido.Length - 1))

End If

Next

Para la opción de filtrar no encontré donde ponerla, ni cómo hacerla para que quedase como yo quería, al final me decidí por poner un TextBox en el ToolStrip y que por cada letra introducida buscara carpetas o ficheros que contengan la frase actual del TextBox. Al crear el método, no aceptaba bien las cadenas de menos de 3 letras, así que puse de tope ese número, a partir del 3 carácter busca los archivos y los va listando en el listBox activo.



La opción buscar me dio bastantes problemas, ya que en un principio iba a ser de una forma y luego la cambié parcialmente. En un principio, debería buscar en todo el ordenador, como hace el buscador normal de Windows, pero una vez terminado, haciendo pruebas descubrí que podía ser muy muy lento y costoso de ejecutar, que podía tardar perfectamente 2 minutos o más. Ya que mi propósito era que el usuario pudiese elegir a grandes rasgos donde buscar ya que siempre sabes por donde está tu fichero, pero no en la carpeta que esta. Dicho esto la funcionalidad de buscar quedó en gran resultado, permitiendo realizar búsquedas de todo el ordenador (si se ejecuta la orden desde la raíz) o en carpetas específicas, como podría ser el Escritorio. De esta forma no todas las búsquedas van a ser tan costosas, si sabes que tu archivo se encuentra en el escritorio pero no sabes en que carpeta, basta con examinar solo el escritorio y por otro lado si no sabes dónde está, puedes examinar el disco para encontrarlo.

La última decisión que tomé con respecto a la búsqueda, es mostrar la ruta entera donde se encuentra el archivo, para así poder saber dónde está. De todas formas el usuario puede hacer doble clic en las carpetas para mostrar su contenido o en un archivo para poder ejecutarlo esté donde esté.

Otras opciones que me parecieron interesantes fueron las de imprimir y mandar un archivo por correo electrónico. En imprimir tuve un par de problemas y es que buscando por internet la forma que más limpia y más sencilla de implementar me pareció fue la de llamar al proceso de impresión de un archivo y que él se encargara de todo. Ej: Queremos imprimir un archivo .odt, nuestro proceso llama a la acción de imprimir asignada a esa extensión, en mi caso lo imprimiría con el open office y gracias a la opción MyProcess.StartInfo.CreateNoWindow = False

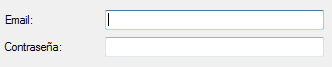
Solo nos mostraría la pantalla de carga del programa (si la tuviera) y luego ejecutaría la impresión directamente sin mostrar nada más. Con esta opción también decidí implementar la multiSelección ya que me parece de gran ayuda poner a imprimir 3 documentos que necesites sin tener que hacerlo uno por uno. El pequeño problema con este método fue que si un archivo no tiene asignado un “proceso” de impresión por ejemplo, un archivo .mp3 ya que ningún programa tiene la opción implementada. En ese caso se ejecuta la acción auxiliar de imprimir un fichero en modo texto que simplemente lee el texto línea a línea y lo manda a la impresora utilizando el objeto que Visual Basic nos proporciona para imprimir: Dim impresora As Printer = New Printer.

El problema de este método es que la impresión sale sin formato ninguno, con lo cual le di como mínimo unos bordes arriba y a la izquierda.

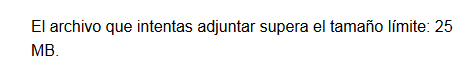
Para mandar Emails con archivos adjuntos tuve que buscar bastante por internet hasta encontrar como se hacía y que funcionara bien. Este paso fue uno de los más “costosos” ya que supone validar campos, los campos requeridos, que el archivo exista etc… Para validar el Email utilice una expresión regular que funciona bastante bien:

Dim re As Regex = New Regex("^([0-9a-zA-Z]([-\.\w]\*[0-9a-zA-Z])\*@([0-9a-zA-Z][-\w]\*[0-9a-zA-Z]\.)+[a-zA-Z]{2,9})$")

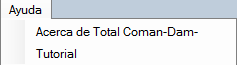
Para mandar un email necesitamos entrar con nuestra cuenta de Email, ya sea Hotmail, Gmail o Yahoo! que son los más usados actualmente según el estudio de ComsCore [3]. Estos servicios tienen una capacidad máxima de archivos que nos limita a 25MB por lo tanto también tendremos que comprobar el tamaño del fichero antes de mandarlo y si es mayor de ese tamaño avisar al usuario. Decidí que sería mejor pedir el usuario y contraseña en ese mismo formulario así no viajan entre ningún formulario dándole más seguridad.



Esta acción funciona con varios archivos siempre que no superen los 25MB de capacidad que nos ofrece el servicio de Email, contra ese problema no pude hacer nada.



Para finalizar pensé que estaría bastante bien crear un tutorial por si alguien no sabe cómo funciona explicarle a grandes rasgos algunas características. Cuando un nuevo usuario inicia la aplicación se inicia el formulario y no se vuelve a mostrar a menos que lo hagamos manualmente desde el menú ayuda.



En las partes que más problemas tuve fue en la de crear el instalador y las pruebas ya que no las vimos en gran profundidad en clase.

Al final no acabé ninguna de las dos debido a que no tendría tiempo suficiente para probar el programa manualmente y arreglar los posibles fallos.

En la parte de realización de pruebas tuve bastantes problemas, ya que no me dejaba importar el proyecto a la solución de pruebas para probar los métodos. Al final lo conseguí importar bien. Al final la deje de lado, ya que por falta de tiempo no podría crear las suficientes pruebas para tener una cobertura de código de entre el 60% y el 80% que es lo recomendable.

**Conclusiones:**

La realización del proyecto me ha parecido bastante buena, estoy bastante orgulloso al ver el programa “acabado” aunque sea un programa bastante pequeño cumple las expectativas que tenía. Creo que la elección de Visual Basic y su IDE Visual Studio fueron un acierto completo ya que con tantas facilidades por parte del lenguaje y del IDE se hace muy cómodo trabajar. Los objetivos logrados han sido los siguientes:

* Manejar los diferentes paneles
* Ejecutar archivos o navegar por carpetas en los paneles
* Copiar archivos de un panel a otro, tanto con Drag&Drop como con los botones de opciones
* Mover archivos de un panel a otro
* Borrar archivos
* Renombrar varios archivos al mismo tiempo
* Comprimir carpetas
* Descomprimir carpetas
* Ejecutar las tareas pesadas en segundo plano
* Utilizar rutas favoritas y todo su manejo
* Crear carpetas y ficheros
* Obtener información del PC y del archivo
* Permitir cambiar la extensión de un archivo
* Imprimir
* Mandar archivos por correo electrónico
* Buscar en subcarpetas
* Filtrar el contenido de una carpeta
* Tutorial
* Permitir algo de personalización al programa con los colores y las letras
* Comparar dos carpetas
* Ordenar por nombre o por fecha
* Arreglar Errores

Unos posibles trabajos futuros son:

* Cambiar el ListBox por un ListView
* Encriptar ficheros
* Mejorar la interfaz
* Creación de varias fichas para paneles

**Bibliografía:**

[1] - <http://www.infoworld.com/slideshow/80986/infoworlds-2013-technology-of-the-year-award-winners-210419#slide12>

[2] - <http://www.comoprogramar.org/ranking-estadisticas-lenguajes-de-programacion-2013/>

[3] - <http://www.siliconnews.es/wp-content/uploads/2012/11/2012-10-31-image-7-500x276.png>