En caso que el origen de un fichero, sea una ruta con la carpeta y el directorio, será preciso utilizar los métodos para separar la información de fichero y de ruta del directorio.

Las clases que se utilizarán son las siguientes contenidos en el espacio de nombres **System.IO**:

- System.IO.File
- System.IO.Directory
- System.IO.FileInfo
- System.IO.DirectoryInfo

COPIAR

Para ejecutar este ejemplo, hay que crear los siguientes directorios y ficheros:

C:\Users\Public\TestFolder
C:\Users\Public\TestFolder\test.txt
C:\Users\Public\TestFolder\SubDir\test.txt

```
static void Main()
{          string NombreFich = "test.txt";
           string Pathorigen = @"C:\Users\Public\TestFolder";
           string PathDestino =  @"C:\Users\Public\TestFolder\SubDir";
           string FichOrigen = System.IO.Path.Combine(Pathorigen, NombreFich);
           string FichDestino = System.IO.Path.Combine(PathDestino, NombreFich);

        // Copiar los contenidos de una carpeta a una localización nueva: Crea una carpeta si es
      necesario.
           if (!System.IO.Directory.Exists(PathDestino))
                {System.IO.Directory.CreateDirectory(PathDestino); }


      // copiar los ficheros a otra localización y sobreescribe el fichero destino si ya existe.
      System.IO.File.Copy(FichOrigen, FichDestino , true);

       // Obtener los archivos en el directorio  y copiar todos los ficheros de un directorio a otro
      directorio. origen .
      if (System.IO.Directory.Exists(PathOrigen))
      {
           string[] files = System.IO.Directory.GetFiles(PathOrigen);
       // Copiar los ficheros y sobrescribe si ya existe .
           foreach (string s in files)
           {
                fileName = System.IO.Path.GetFileName(s);
                FichDestino  = System.IO.Path.Combine(PathDestino, fileName);
                System.IO.File.Copy(s, FichDestino , true);
           }
      }
      else
      {
           Console.WriteLine("Path origen no existe!");
      }
      Console.WriteLine("Pulsar cualquier tecla para salir.");
Console.ReadKey();
}
```

Hacer especial hincapie, en que cuando le pasamos archivo de origen y archivo de destino, para crear los dos valores, tenemos que crearlos con la combinación de directorio y archivo.

MOVER

```
string FichOrigen = @"C:\Users\Public\public\test.txt";
string destinationFile = @"C:\Users\Public\private\test.txt";

        // Mover el fichero a otra localización:
                System.IO.File.Move(FichOrigen, destinationFile);

        // Mover un directorio modificando o combinando su contenido
                System.IO.Directory.Move(@"C:\Users\Public\public\test\",
                @"C:\Users\Public\private");
```

ELIMINAR

Crear los siguientes directorios

C:\Users\Public\DeleteTest\test1.txt
C:\Users\Public\DeleteTest\test2.txt
C:\Users\Public\DeleteTest\SubDir\test2.txt

```
// borrado utilizando un método Static
    if(System.IO.File.Exists(@"C:\Users\Public\DeleteTest\test.txt"))
        {
        try
        {
                System.IO.File.Delete(@"C:\Users\Public\DeleteTest\test.txt");
        }
        catch (System.IO.IOException e)
        {
                Console.WriteLine(e.Message);
                return;
        }
}
```

```
// O bien  borrando usando la clase  using FileInfo  instanciandola
    System.IO.FileInfo fi = new System.IO.FileInfo(@"C:\Users\Public\DeleteTest\test2.txt");
    try
    {
        fi.Delete();
    }
    catch (System.IO.IOException e)
    {
        Console.WriteLine(e.Message);
    }
```

```
// Borrado de un directorio, el cual debe tener permiso de escritura o estar vacío.

try
{
        System.IO.Directory.Delete(@"C:\Users\Public\DeleteTest");
}
catch (System.IO.IOException e)
{
        Console.WriteLine(e.Message);
}
}
```

```
// Borrar un directorio y todos sus directorios con un método Static…
if(System.IO.Directory.Exists(@"C:\Users\Public\DeleteTest"))
{
try
{
System.IO.Directory.Delete(@"C:\Users\Public\DeleteTest", true);
}
catch (System.IO.IOException e)
{
Console.WriteLine(e.Message);
}
```

```
// o con  DirectoryInfo In tanciandola
System.IO.DirectoryInfo di = new System.IO.DirectoryInfo(@"C:\Users\Public\public");
try
{
        di.Delete(true);
}
catch (System.IO.IOException e)
{
        Console.WriteLine(e.Message);
}
```

Al igual que nos puede interesar utilizar un dialogo para seleccionar un archivo concretamente, también nos puede interesar seleccionar una carpeta o directorio.

```
FolderBrowserDialog dialogoRuta=new FolderBrowserDialog();

 if (dialogoRuta.ShowDialog()==DialogResult.OK)

{testbox.text=dialogoRuta.SelectedPath;

 }
```

Método **.GetDirectoryName** contenido en System.IO.Path que devuelve la ruta del directorio sin el nombre del fichero.

```
string Directorio=System.IO.Path.GetDirectoryName(nombreFich);
```

Método **.GetFileName** contenido en System.IO.Path que devuelve el nombre del fichero

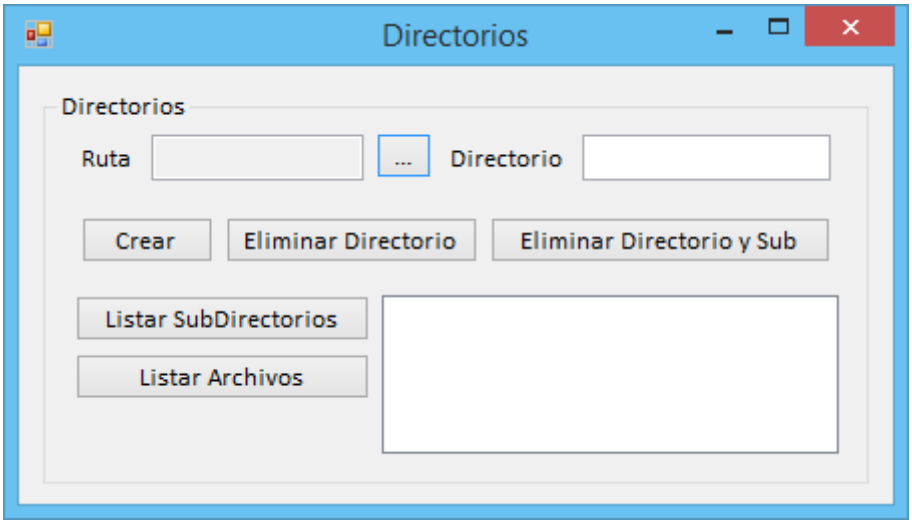
```
string Archivo=System.IO.Path.GetfileNameName(nombreFich);
```

Ejercicio

Realizar un programa para crear, eliminar y listar directorios. Ademas tendrá la capacidad de listar archivos y eliminar todo el contenido de un directorio.

Interfaz Gráfica

La interfaz gráfica es la siguiente:



Todo esto va ser posible gracias a la clase que nos provee el .NET Framework llamada **Directory** que se encuentra en el espacio de nombres **System.IO**

Debemos importar nuestra referencia para obtener acceso a dicha clase

```
using System.IO;
```

Crear Directorio

Vamos a comenzar por crear un directorio, esto es posible con el método **CreateDirectory** en el cual debemos pasar como argumento la ruta de la ubicación de nuestro directorio junto al nombre de nuestro nuevo directorio. Es por eso que en nuestra interfaz tenemos un **FolderBrowserDialog** (...) que nos sirve para buscar la ubicación y el textbox directorio en el cual vamos a indicar el nombre de nuestro nuevo directorio.

Para crear nuestro nuevo directorio

- Se harán validaciones para evitar que pinche nuestro programa, de hecho podríamos poner un try catch y lanzar una excepción personalizada,

- Primeros validamos si nuestro ubicación esta vacía utilizando el método IsNullOrEmpty, si es así, significa que no hemos seleccionado ninguna ubicación y mostramos un mensaje.

- Si hemos seleccionado una ubicación, debemos validar si existe (puede ocurrir que otro usuario la elimine), es por eso que utilizamos el método Exist de nuestra clase **Directory** en el cual pasamos como argumento la ubicación del directorio.

- Si todo va bien, llego el momento de crear nuestro directorio con el método **CreateDirectory** donde le pasamos como argumento la ubicación junto al nombre de nuestra carpeta.

Eliminar Directorio

Para eliminar un directorio vamos a utilizar el método **Delete** en el cual pasamos como argumento la ruta de nuestro directorio. Ademas en este método posee otra sobrecarga el cual tiene como argumento la ruta de nuestro directorio y un valor boolean que especifica si se borran todos los subdirectorios y archivos del directorio.

Como este método borra un directorio que este vació por completo, debemos realizar una serie de validaciones.

Primero validamos si hay una ruta y ademas si posee algún directorio dicha ruta. En caso de pasar esa validación, validamos si el directorio posee uno o mas archivos, si no posee procedemos a eliminar el directorio con el método **Delete.** También podemos validar si ese directorio existe con el método **Exist** .Este método con el argumento de tipo boolean en **true** elimina todo el contenido sin importar lo que haya adentro.

Listar SubDirectorios y Archivos

En este caso vamos a ver como poder listar los subdirectorios que tiene un directorio. Esto es posible gracias al método GetDirectories que devuelve un array de string.

Solo tenemos que recorrer el array y agregar los items en un listbox

Declaramos un array de string llamado directorios y lo asignamos al metodo **GetDirectories** el cual le pasamos como argumento nuestra ruta del directorio.

Recorremos el array y vamos agregando cada item a nuestro listbox.

Lo mismo ocurre para listar archivos dentro de un directorio, en este caso, se divide en dos partes, ya que primero vamos a listar los archivos en el directorio actual y por otra parte vamos a listar los archivos en los subdirectorios

Lo mismo ocurre en **GetFiles** devuelve una matriz de string que contiene las rutas de los archivos, donde también agregamos los archivos a nuestro listbox.


CIFP
Centros
Integrados
Formación Profesional