Este proyecto es un buscador rustico, el cual devuelve una serie documentos ordenados por importancia en dependencia de un query introducido por un usuario.

El primer paso para lograr esto es ejecutar un preprocesamiento de la base de datos de documento de texto. Esto se hace en el programa creando una instancia de la clase Corpus. Esta clase esta caracterizada por:

- Una matriz de valores tipo double, con cada fila correspondiente a un documento y cada columna a una palabra, almacenando en cada indice el valor de TF*IDF de cada palabra en cada documento.
  (double[,] TfIDFMatrix)

- Un vector con todas las palabras del corpus y su valor de IDF.
  (string[] WordVector)

- Un array de strings que contiene los Paths de cada documento del corpus.
  (string[] DocPaths)

- Un string correpondiente al Path de la carpeta donde esta almacenada la base de datos.
  (string FilePath)

Al ejecutar el programa se crea esta instancia, cargando y procesando los documentos. A continuacion se recibe el query y se procesa como un vector en funcion de esta instancia de la clase, poniendo el valor del IDF en los indices de las palabras pertenecientes al query y 0 en el resto de los indices. Luego halla el valor de score de cada documento respecto al query con el metodo de "similitud de coseno" que expresa la cercania entre dos vectores de un espacio como un numero entre 0 y 1 y cuyo metodo de obtencion seria :

```
public double[] VectorProduct(double[] queryVector)
{
 double[,] Matrix = TfIDFMatrix;
  double[] output = new double[Matrix.GetLength(0)];
    for (int i = 0; i < Matrix.GetLength(0); i++)
   {
     for (int j = 0; j < queryVector.Length; j++)
     {
       if(query[j] == 0)
          {
            continue;
          }
         output[i] += Matrix[i,j]*queryVector[j];
      }
    }
 return output;
 }
```

Tenemos ya el array de scores de cada documento obtenido con este metodo. Construimos otro array que contenga los titulos de cada documento, utilizando el metodo GetTitulos y un array mas conteniendo un snippet de cada texto en correspondencia con la palabra mas relevante del query con el metodo CreateSnippets.

Para finalizar es necesario ordenar estos arrays para tener cada documento segun su relevancia con la query:

```
    public static void OrdenandoArrays(double[] values, string[] titulos,
    string[] snippets)
{
    List<Tuple<double, string, string>> listaTuplas = new List<Tuple<double,
    string, string>>();

    for (int i = 0; i < valores.Length; i++)
    {
        listaTuplas.Add(new Tuple<double, string, string>(values[i], titulos[i],
        snippets[i]));
    }


    listaTuplas.Sort(); // Ordenar la lista de tuplas en función del valor del
        primer array


    for (int i = 0; i < listaTuplas.Count; i++)// Extraer los valores de los otros
        dos arrays en el orden correcto
    {
        values[i] = listaTuplas[i].Item1;
        titulos[i] = listaTuplas[i].Item2;
        snippets[i] = listaTuplas[i].Item3;
    }
}
```

Este metodo devuelve todos los arrays ordenados con lo cual solo restaria introducir cada elemento de estos en un objeto tipo SearchItem y retornarlos como resultado del metodo Query