Informe del Primer Proyecto de Programación … Moogle!

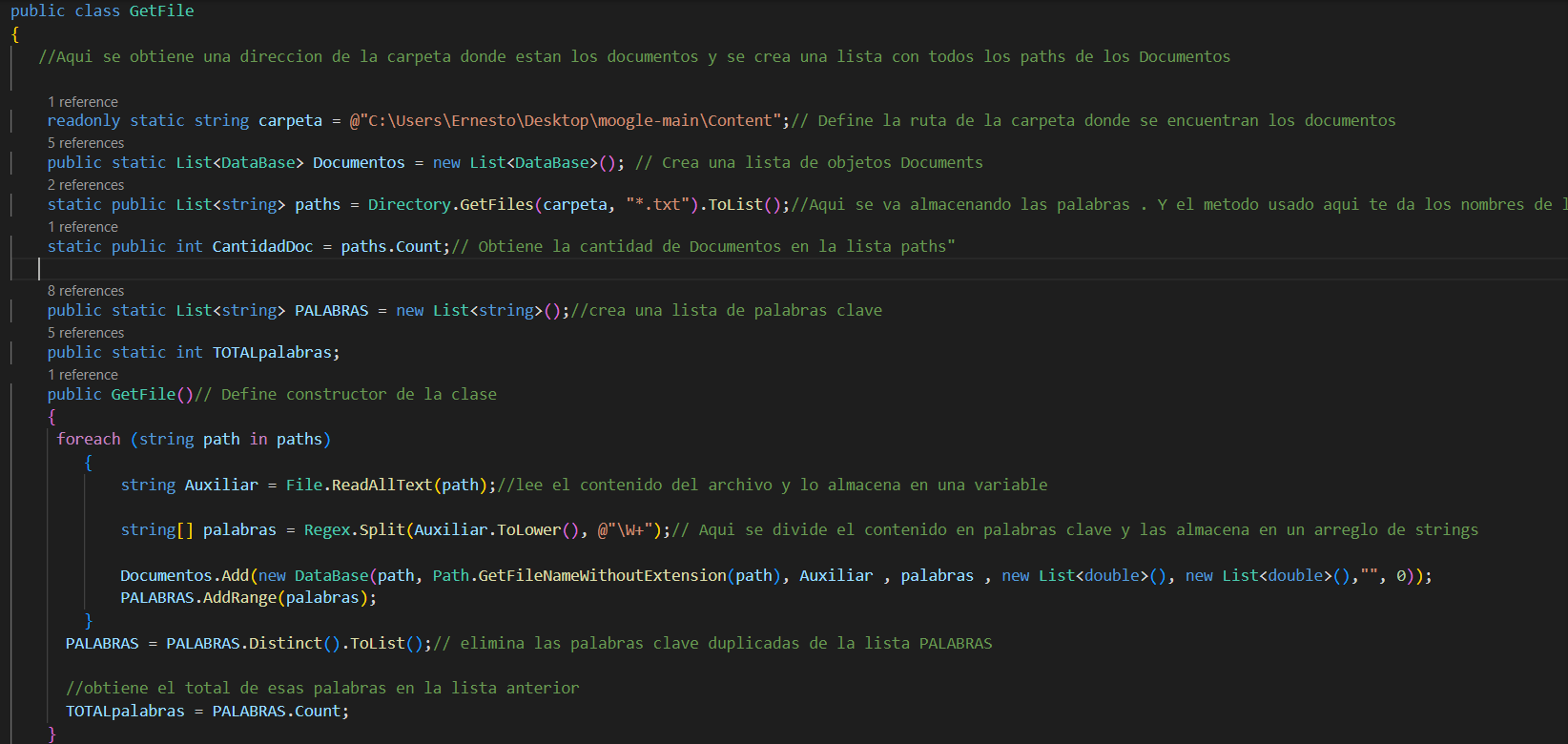
Ernesto Javier Govea Varona CC121

Moogle! es un motor de búsqueda con el objetivo de buscar un texto en un conjunto de documentos. A continuación se expondrá de manera resumida algunas funcionabilidades del mismo :

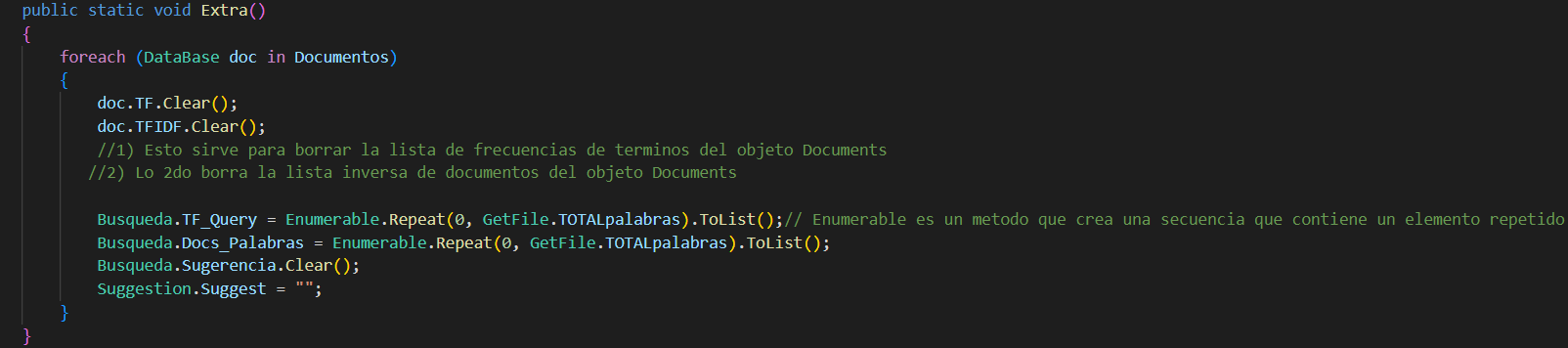
1) En primer lugar se necesitaba indexar los documentos y en esto consiste la clase "GetFile" que se encarga de obtener los documentos de una carpeta, procesarlos y almacenarlos en una lista de objetos "DataBase". También se encarga de obtener las palabras clave de los documentos y almacenarlas en una lista. El método "Extra" se utiliza para borrar las listas de frecuencias de términos y de documentos inversos de cada objeto "DataBase" en la lista "Documentos".

.La propiedad "carpeta" define la ruta de la carpeta donde se encuentran los documentos que se van a procesar. La propiedad "Documentos" es una lista de objetos "DataBase" que representan los documentos que se van a procesar. La propiedad "paths" es una lista de rutas de archivo que se obtienen de la carpeta definida en "carpeta". La propiedad "PALABRAS" es una lista de palabras clave que se obtienen de los documentos.

.El constructor de la clase "GetFile" se encarga de leer el contenido de cada archivo en la lista "paths", dividir el contenido en palabras clave y almacenarlas en la lista "PALABRAS". También crea un objeto "DataBase" para cada archivo y lo agrega a la lista "Documentos".



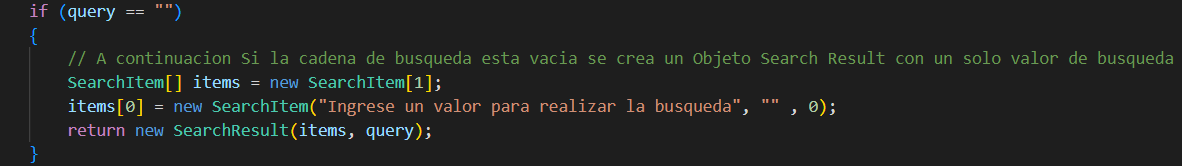
* Método Extra



Al estar funcionando el programa y el usuario introduce una búsqueda sucede que :

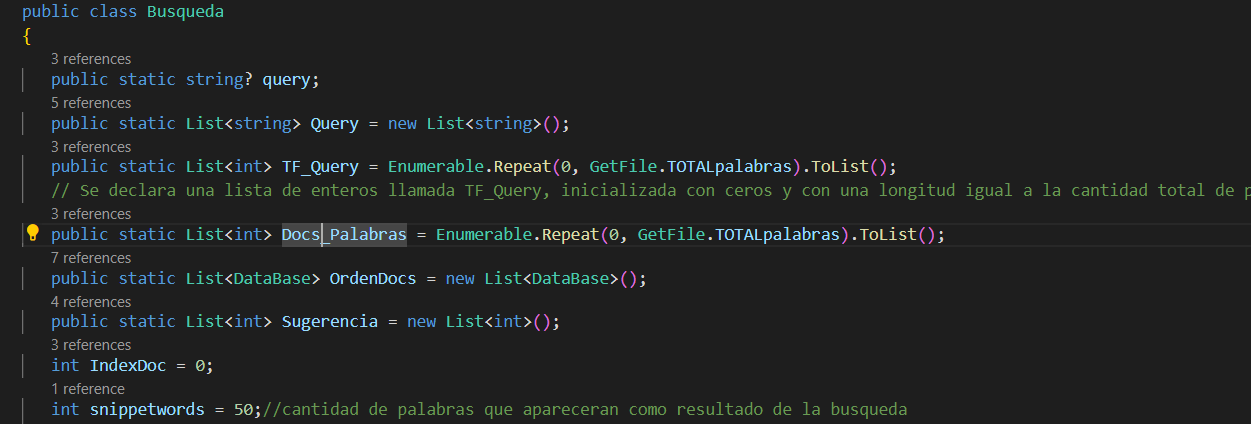
. Se llama a un método el cual permite regresar al estado inicial y poder realizar varias búsquedas sin la necesidad de reiniciar el programa.

\*Si la búsqueda es vacía se devuelve un SsearchResult de esta manera :



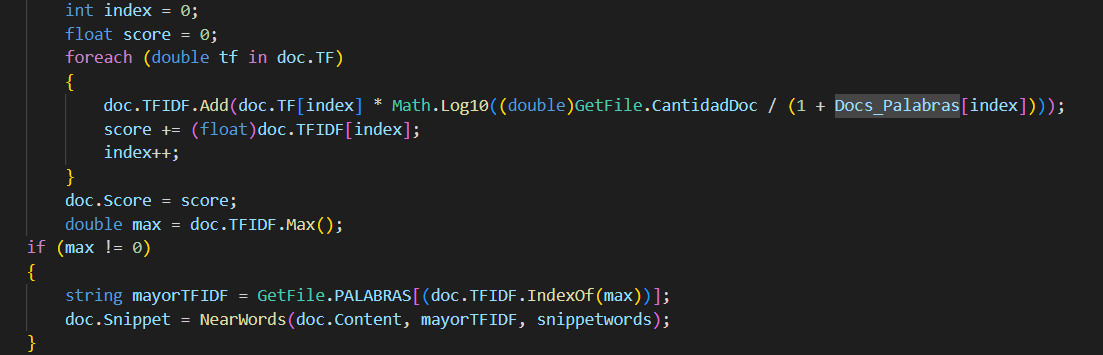
* Clase Búsqueda

La propiedad "query" es una cadena que representa la consulta de búsqueda. La propiedad "Query" es una lista de cadenas que se obtiene al dividir la consulta en palabras clave utilizando una expresión regular. La propiedad "TF\_Query" es una lista de enteros que representa la frecuencia de términos de la consulta. La propiedad "Docs\_Palabras" es una lista de enteros que representa la cantidad de documentos que contienen cada palabra clave de la consulta. La propiedad "OrdenDocs" es una lista de objetos "DataBase" que representa los documentos ordenados por puntaje de relevancia.

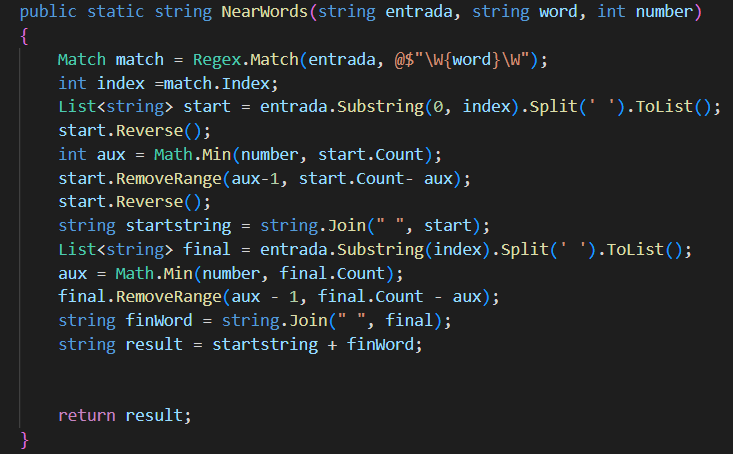


El constructor de la clase "Busqueda" se encarga de procesar la consulta de búsqueda y calcular el puntaje de relevancia de cada documento. Primero, divide la consulta en palabras clave y calcula la frecuencia de términos de la consulta. Luego, para cada palabra clave de la consulta, cuenta la cantidad de veces que aparece en cada documento y calcula la frecuencia de términos de cada documento.

A continuación, calcula el puntaje de relevancia de cada documento utilizando la fórmula TF-IDF. Finalmente, ordena los documentos por puntaje de relevancia y los almacena en la lista "OrdenDocs".



.Método NearWords



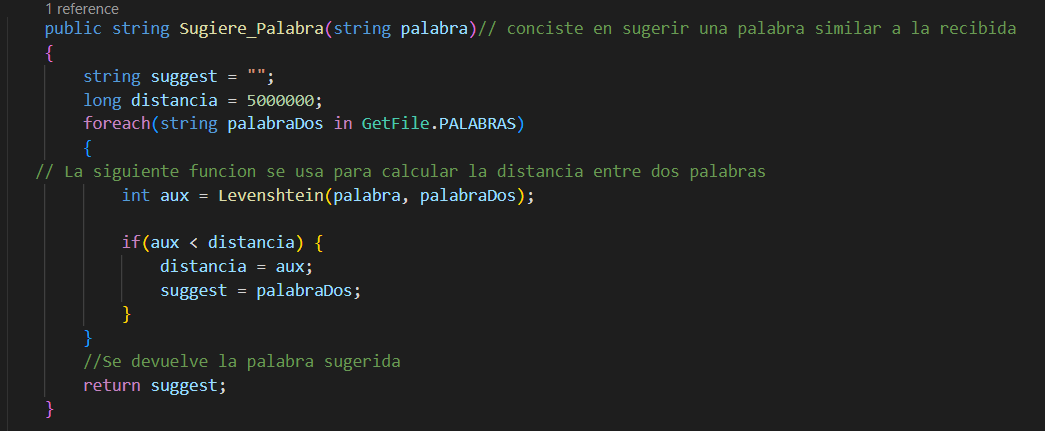
Este método se encarga de encontrar las palabras cercanas a la palabra clave más relevante de un documento y devuelve una cadena que representa un fragmento del del documento.

* Clase Suggest

En resumen esta clase consiste en que si las palabras de la búsqueda se encuentran o no en los documentos , le sugiere al usuario a través de la distancia Levenshtein las palabras que más se asemejan a su búsqueda .

La clase tiene dos métodos: "Levenshtein" y "Sugiere\_Palabra". El método "Levenshtein" recibe dos cadenas de texto y devuelve un entero que representa la distancia de Levenshtein entre las dos cadenas.

El método "Sugiere\_Palabra" recibe una cadena de texto y devuelve una cadena de texto que representa una palabra sugerida similar a la recibida. Para ello, se compara la cadena recibida con todas las palabras almacenadas en la lista "PALABRAS" de la clase "GetFile" utilizando el método "Levenshtein" para calcular la distancia de Levenshtein entre la cadena recibida y cada palabra de la lista. Se devuelve la palabra de la lista que tenga la menor distancia de Levenshtein con la cadena recibida.



El constructor de la clase "Suggestion" se encarga de recorrer la lista "Sugerencia" de la clase "Busqueda" y, para cada valor de la lista, si el valor es 0, llama al método "Sugiere\_Palabra" con el valor que tiene la lista "Query" en la posición del índice actual. Luego, agrega el valor actual de "Query" a la variable "Suggest". Al finalizar el recorrido de la lista "Sugerencia", se reinicia el valor del índice a 0

