

PROYECTO DE PROGRAMACIÓN I

Claudia Hernández Pérez
C-122

Facultad de Matemática y Computación

20 de julio de 2023

Moogle!

Claudia
Hernández
Pérez

Moogle!

Claudia
Hernández
Pérez

Moogle! es una aplicación totalmente original cuyo propósito es buscar inteligentemente un texto en un conjunto de documentos.

Es una aplicación web, desarrollada con tecnología .NET Core 7.0, específicamente usando Blazor como framework web para la interfaz gráfica, y en el lenguaje C sharp.

La aplicación esta dividida en dos componentes fundamentales:

Moogle!

Claudia
Hernández
Pérez

Moogle! es una aplicación totalmente original cuyo propósito es buscar inteligentemente un texto en un conjunto de documentos.

Es una aplicación web, desarrollada con tecnología .NET Core 7.0, específicamente usando Blazor como framework web para la interfaz gráfica, y en el lenguaje C sharp.

La aplicación esta dividida en dos componentes fundamentales:

- MoogleServer es un servidor web que renderiza la interfaz gráfica y sirve los resultados.

Moogle!

Claudia
Hernández
Pérez

Moogle! es una aplicación totalmente original cuyo propósito es buscar inteligentemente un texto en un conjunto de documentos.

Es una aplicación web, desarrollada con tecnología .NET Core 7.0, específicamente usando Blazor como framework web para la interfaz gráfica, y en el lenguaje C sharp.

La aplicación esta dividida en dos componentes fundamentales:

- MoogleServer es un servidor web que renderiza la interfaz gráfica y sirve los resultados.
- MoogleEngine es una biblioteca de clases donde está implementada la lógica del algoritmo de búsqueda.

Moogle!

Claudia
Hernández
Pérez

Esta clase se encarga de procesar los documentos de la base de datos antes de comenzar la búsqueda. Se llama al método en la línea anterior a `app.Run ()` para realizar esta operación una sola vez antes de que arranque el programa y esté listo para ejecutarse.

Moogle!

Claudia
Hernández
Pérez

Esta clase se encarga de procesar los documentos de la base de datos antes de comenzar la búsqueda. Se llama al método en la línea anterior a `app.Run()` para realizar esta operación una sola vez antes de que arranque el programa y esté listo para ejecutarse.

```
26  MoogleEngine.ProcessDocuments.LoadDocuments();  
27  app.Run();
```

Moog!e!

Claudia
Hernández
Pérez

Esta clase esta dirigida al trabajo con los documentos, se “normalizan”. Es decir, en caso de que las búsquedas se realicen en español, se eliminan las tildes para evitar errores ortográficos. Además, independientemente del idioma, se eliminan los signos de puntuación y cualquier otro símbolo ajeno al alfabeto y los números.

Esta clase esta dirigida al trabajo con los documentos, se “normalizan”. Es decir, en caso de que las búsquedas se realicen en español, se eliminan las tildes para evitar errores ortográficos. Además, independientemente del idioma, se eliminan los signos de puntuación y cualquier otro símbolo ajeno al alfabeto y los números.

```
8      // Función para normalizar los textos y hacer mas cómoda la búsqueda evadiendo errores de ortografía y símbolos innecesarios
9      2 references
      public static string Normalize(string text)
10     {
11         text = text.ToLower();
12
13         text = text.Replace("á", "a");
14         text = text.Replace("é", "e");
15         text = text.Replace("í", "i");
16         text = text.Replace("ó", "o");
17         text = text.Replace("ú", "u");
18
19         text = Regex.Replace(text, @"[^\a-zñ0-9]", " ");
20
21         return text;
22     }
```

Moog!e!

Claudia
Hernández
Pérez

Se encuentra además esta otra función para seleccionar el “snippet” (pedazo breve de un texto) que se imprimirá en el momento de la búsqueda por cada resultado.

Se encuentra además esta otra función para seleccionar el “snippet” (pedazo breve de un texto) que se imprimirá en el momento de la búsqueda por cada resultado.

```
24 public static string Snippet(string text, string textwhitoutnormalize, string word) {
25     //text texto normalizado
26     //textwhitoutnormalize texto sin normalizar
27     //word palabra con mayor IDF en el texto
28
29     string? snippet = null;
30
31     int index = text.IndexOf(word); // Índice de la primera ocurrencia de la secuencia de caracteres de la palabra
32     while (snippet == null) {
33         if ((index == 0
34             || text.Substring(index - 1, 1) == " "
35             && (index + word.Length == text.Length
36             || text.Substring(index + word.Length, 1) == " ")) // Verificar que sea exactamente la palabra
37         {
38             if (index < 150) { // Imprime los primeros caracteres hasta la primera ocurrencia de la palabra
39                 snippet = textwhitoutnormalize.Substring(0, index);
40             } else if (index > 150) { // Imprime 150 caracteres antes de la primera ocurrencia de la palabra hasta esta
41                 snippet = textwhitoutnormalize.Substring(index - 150, 150);
42             }
43             // Concatena la otra mitad del snippet
44             if (((text.Length - 1) - index) < 150) { // Imprime desde la palabra hasta el final del documento
45                 snippet += textwhitoutnormalize.Substring(index);
46             } else if (((text.Length - 1) - index) > 150) { // Imprime desde la palabra y 150 caracteres después
47                 snippet += textwhitoutnormalize.Substring(index, 150);
48             }
49         }
50         // Si llega aquí es porque no era exactamente la palabra y se le indica que continúe hasta la próxima ocurrencia
51         index = text.IndexOf(word, index + 1);
52     }
53 }
54
55 return snippet;
56 }
```

Moog!e!

Claudia
Hernández
Pérez

El valor de “relevancia” de una palabra está dado por el cálculo de su TF (Term Frequency) por su IDF (Inverse Document Frequency), para ello se ha utilizado la fórmula:

$$\frac{nd}{Cd} \cdot \log\left(\frac{T}{N}\right)$$

Dónde:

- nd es la cantidad de ocurrencias de una palabra en un documento,

Moog!e!

Claudia
Hernández
Pérez

El valor de “relevancia” de una palabra está dado por el cálculo de su TF (Term Frequency) por su IDF (Inverse Document Frequency), para ello se ha utilizado la fórmula:

$$\frac{nd}{Cd} \cdot \log\left(\frac{T}{N}\right)$$

Dónde:

- nd es la cantidad de ocurrencias de una palabra en un documento,
- Cd es la cantidad total de palabras en el documento,

Moog!e!

Claudia
Hernández
Pérez

El valor de “relevancia” de una palabra está dado por el cálculo de su TF (Term Frequency) por su IDF (Inverse Document Frequency), para ello se ha utilizado la fórmula:

$$\frac{nd}{Cd} \cdot \log\left(\frac{T}{N}\right)$$

Dónde:

- nd es la cantidad de ocurrencias de una palabra en un documento,
- Cd es la cantidad total de palabras en el documento,
- T es la cantidad total de documentos,

Moog!e!

Claudia
Hernández
Pérez

El valor de “relevancia” de una palabra está dado por el cálculo de su TF (Term Frequency) por su IDF (Inverse Document Frequency), para ello se ha utilizado la fórmula:

$$\frac{nd}{Cd} \cdot \log\left(\frac{T}{N}\right)$$

Dónde:

- nd es la cantidad de ocurrencias de una palabra en un documento,
- Cd es la cantidad total de palabras en el documento,
- T es la cantidad total de documentos,
- N es la cantidad de documentos en los que aparece la palabra.

Moogle!

Claudia
Hernández
Pérez

Esta es la clase principal del programa, donde comienza el proceso de búsqueda. Comienza en el momento en que se recibe la query.

Moogle!

Claudia
Hernández
Pérez

Esta es la clase principal del programa, donde comienza el proceso de búsqueda. Comienza en el momento en que se recibe la query.



Moogle!



Buscar

Mooglee!

Claudia
Hernández
Pérez

Se iteran las palabras sin repetir de la query y se calcula su TF-IDF en los documentos en los que aparece, pero para conseguir el score por documento se necesita la suma de estos valores.

El score queda de la siguiente manera: si la palabra está contenida en el documento que se está analizando se encontrará el valor de TF-IDF correspondiente a esa palabra en ese documento, de forma contraria el valor será 0.

Moogle!

Claudia
Hernández
Pérez

Adicionalmente, implementé que al devolver los resultados de la búsqueda imprimiera el tiempo que había tardado la búsqueda con el texto en pantalla:

Moogle!

Claudia
Hernández
Pérez

Adicionalmente, implementé que al devolver los resultados de la búsqueda imprimiera el tiempo que había tardado la búsqueda con el texto en pantalla:

La búsqueda demoró **0.0391181** segundos

Moogle!

Claudia
Hernández
Pérez

Asimismo, se puede buscar dando click en el botón buscar o presionando la tecla “enter”. Además de imprimir en pantalla los resultados de la búsqueda:

Moog!e!

Claudia
Hernández
Pérez

Asimismo, se puede buscar dando click en el botón buscar o presionando la tecla “enter”. Además de imprimir en pantalla los resultados de la búsqueda:

Se encontraron **29** resultados relacionados con la búsqueda

Moogle!

Claudia
Hernández
Pérez

Moogle!

 **Buscar**

¿Quisiste decir [programming](#)?

La búsqueda demoró **0.0391181** segundos

Se encontraron **29** resultados relacionados con la búsqueda

Moogle!

Claudia
Hernández
Pérez

Si es de su interés, puede encontrar un informe más detallado del proyecto en este link:

<https://github.com/ClaudiaHdezPerez/moogle-project/tree/main/Informe>

Aquí puede encontrar el código del programa y ejecutarlo con el comando `dotnet watch run --project MoogleServer` en la terminal de Visual Studio Code:

<https://github.com/ClaudiaHdezPerez/moogle-project/tree/main/MoogleEngine>